BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
Facultad de Ciencias Físico Matemáticas

Principales Métodos de Aprendizaje Automático y su Aplicación: Un Caso Práctico con Datos de la Fundación JUCONI.

PRESENTA
Andrés Aram De la Calleja Macip

DIRECTOR DE TESIS
Dr. Bulmaro Juárez Hernández

TESIS PARA OBTENER EL GRADO EN:
LICENCIATURA EN ACTUARÍA

Octubre-2024

A mi familia, amigos y a mi yo de seis años,
una parte de ustedes habita aquí.
Gracias por hacer mi vida maravillosa.
-Aram.

Pueden llamarlo transformación. Metamorfosis. Falsedad. Traición.
Yo lo llamo ***Una Educación****.*
- Tara Westover, Una Educación.

Índice general

Agradecimientos

Hoy, en el momento en que concluye esta etapa de mi vida académica y personal, me siento profundamente agradecido y honrado por todas las personas e instituciones que han contribuido de manera significativa a mi desarrollo y logros.

Deseo expresar mi reconocimiento a cada uno de ustedes:

Al Doctor Bulmaro, mi asesor de tesis, le agradezco de todo corazón por su inquebrantable apoyo y orientación a lo largo de este arduo camino. Sus conocimientos, paciencia y dedicación han sido fundamentales en mi formación académica, y estoy eternamente agradecido por su mentoría. La fuerza nos acompaña.

A la Fundación JUCONI, mi fuente de datos y recursos invaluable, quiero expresar mi sincero agradecimiento. Su generosidad al proporcionar la información necesaria para mi investigación ha sido esencial para el éxito de mi proyecto; además de hacerme crecer más como persona al permitirme hacer las practicas con ustedes. Un agradecimiento especial al Ingeniero Gregorio y a la Licenciada Diana, quienes atendieron mis necesidades como estudiante y me brindaron siempre su apoyo.

A mi madre, Rosalía, quien ha sido mi inspiración y el faro de mi vida, no tengo palabras suficientes para agradecer su apoyo inquebrantable, amor y sacrificio. Gracias por ser mi ejemplo, mi pilar de vida y, sobre todo, gracias por ser mi madre.

A mi padre^{+}, Andrés, quien (aunque no esté presente en este mundo terrenal) me sigue enseñando todos los días el significado de la tenacidad y la voluntad.

A mis hermanos, Vanessa y Eduardo, cuyos ánimos y palabras de aliento me han dado fuerzas en los momentos difíciles, les agradezco de todo corazón. Tenerlos de hermanos a ustedes es un regalo inestimable.

A mis abuelitas, (ambas se llaman Socorro), que han brindado su cariño y sabiduría a lo largo de los años, les envío un agradecimiento especial por las lecciones de vida que me han transmitido, por su amor incondicional y por siempre tratarme con amor.

A mi abuelo, Eduardo, quien siempre me impulsó a seguir mis sueños y a esforzarme al máximo, le dedico mi más profundo agradecimiento. Sus enseñanzas siguen siendo mi guía y lo seguirán siendo hasta el final de estos.

A mi familia, tanto cercana como extendida, les agradezco por su apoyo constante, sus palabras de aliento y su amor.

A mis amigos, quienes han compartido risas, lágrimas y experiencias invaluables, les agradezco por su amistad sincera. Su presencia en mi vida ha hecho que cada día sea más significativo. Un agradecimiento especial a Aranza, Lizeth, Daisy, Mónica, Iván, Víctor, Bobby y Yersin; gracias por ser siempre los reales.

A mis sinodales, Dra. Gladdys Salgado, Dra. Hortensia Reyes, M.C. Brenda Zavala que evaluaron mi tesis, les agradezco por su tiempo, esfuerzo y valiosas contribuciones que han enriquecido mi

trabajo y mi aprendizaje.

A la Facultad de Físico-Matemáticas, mi segunda casa durante estos años de estudio, les agradezco por brindarme la oportunidad de adquirir conocimientos y por el ambiente académico enriquecedor que han creado.

A mis amigos de INNOCEAN, donde tengo mi primer trabajo. Gracias por aceptarme e incluirme de manera tan amable y, sobre todo, hacerme sentir en casa. Un agradecimiento especial a Jona y Lalo, gracias por tanto.

En resumen, a todos ustedes, mi gratitud es inmensa. Sin su apoyo, guía y amor, no habría llegado a donde estoy hoy. Cada uno de ustedes ha dejado una huella imborrable en mi vida, y les estaré eternamente agradecido por ello.

Con gratitud y aprecio, Aram.

"May the flames guide thee."

•

Solo los que viven en el presente pueden hacer una nueva época.
Silvers Raleyght, One Piece.

1 Introducción

El concepto de aprendizaje automático, que permite a las máquinas aprender tareas no específicamente programadas y entrenarse en entornos desconocidos, tiene sus raíces en el siglo XVII con la invención de la calculadora mecánica de Blaise Pascal[38]. Este intento pionero de automatizar el procesamiento de datos mediante engranajes y ruedas marcó el comienzo de la búsqueda por dotar a las máquinas de la capacidad de aprendizaje.

En la actualidad, el término aprendizaje automático comenzó a ganar popularidad en la década de 1990, gracias al avance del poder de procesamiento informático que posibilitó la implementación y mejora de procesos de entrenamiento, permitiendo que las máquinas aprendieran y se mejoraran por sí mismas. Aunque las bases teóricas fueron establecidas por Alan Turing en 1936 con su concepto de máquina universal, el término fue formalmente acuñado por Arthur Samuel en 1952[11]. Sin embargo, las limitaciones computacionales restringieron su desarrollo hasta la última década.

Con el auge de la informática y la abundancia de datos, el aprendizaje automático floreció. En la década de 1990, algoritmos más sofisticados y la creación de conjuntos de datos más grandes permitieron un progreso significativo. La introducción de técnicas como Support Vector Machines y Random Forests marcó un hito en la creación de modelos para el análisis, inferencia y proyección de datos.

En la última década, el crecimiento de la inteligencia artificial *(Artificial Intelligence)* AI por sus siglas en inglés y el aprendizaje automático *(Machine Learning)* ML por sus siglas en inglés; se ha acelerado, impulsado por mejoras en las redes neuronales y algoritmos de aprendizaje profundo. Empresas líderes como Google, Facebook y Amazon han liderado el camino, integrando estas tecnologías en productos y servicios cotidianos. Esto hace que estas tecnologías permiten en la sociedad, además de que la creciente influencia del aprendizaje automático y la inteligencia artificial plantea cuestionamientos significativos en el ámbito moral y ético. Por un lado, la automatización de decisiones puede mejorar la eficiencia y optimizar recursos, pero también plantea preocupaciones sobre la equidad y la discriminación. La opacidad de los algoritmos y la falta de explicabilidad en las decisiones automatizadas generan desafíos éticos, ya que es difícil responsabilizar a las máquinas por sus acciones. Además, la recopilación masiva de datos plantea inquietudes sobre la privacidad y la seguridad. La implementación de estas tecnologías requiere consideraciones éticas en la toma de decisiones y la mitigación de posibles consecuencias

negativas, con el objetivo de asegurar que la inteligencia artificial se utilice de manera responsable y respetuosa con los valores fundamentales de la sociedad[20].

En resumen, el aprendizaje automático ha vivido un fascinante viaje desde sus raíces teóricas hasta convertirse en una fuerza impulsora detrás de la revolución digital actual. Su crecimiento continuo promete transformar aún más la forma en que interactuamos con la tecnología y abordamos problemas complejos. En un contexto más específico, el reconocimiento y la identificación de patrones son esenciales en la toma de decisiones de diversos ámbitos. Pero, dentro de esta diversificación, la implementación en ámbitos sociales es, en proporción, menor a las demás. Esto abre un área de oportunidad muy amplia para la investigación al combinar los conocimientos del aprendizaje automático y de la inteligencia artificial. Combinándolas se puede aplicar a problemas, procesos y eventos sociales para diferentes análisis con la ventaja de ser más profundos.

Existe evidencia de que estas tecnologías se aplican de manera efectiva en el ámbito social, tanto en cuestiones aplicadas en lo académico[9] como sus implicaciones e impacto en la sociedad [22].

1.1. Planteamiento

La Fundación Junto con los Niños (JUCONI) A.C. es una organización de la sociedad civil sin fines de lucro debidamente registrada en México, Estados Unidos y Reino Unido. Fue fundada en la Ciudad de Puebla en 1989 cuando Gabriel Benítez, Sarah Thomas de Benítez y Joanna Wright decidieron, con el apoyo del International Children's Trust, crear una organización. El objetivo es brindar atención profesional a través de programas a niñas y niños en situación de calle, en riesgo o expuestos a la violencia familiar [13].

Al trabajar con tantos individuos los datos recabados son masivos. Por ende se pueden encontrar y reconocer patrones dentro de base de datos para el análisis de diferentes rubros dentro de la Fundación.

Es viable analizar cuestiones tanto de índole social como psicológico y hasta geográfico al contener información tan variada.

1.2. Objetivos

Este trabajo tiene por meta un análisis a un caso de estudio en la Fundación JUCONI A.C. El cual consiste en comprender la influencia de variables en subprogramas específicos a través del aprendizaje automático. Esto ofrece un valioso marco de análisis y comparación. Permite dirigir recursos hacia áreas de oportunidad y mejorar los entornos de los subprogramas, utilizando variables como escolaridad, edad e ingresos para predecir la participación de individuos en cada subprograma; facilitando así la planificación y la toma de decisiones informadas. La aplicación de este enfoque promete contribuir significativamente al cuestionamiento y efectividad de los subprogramas en cuestión.

1.3. Metodología

La metodología utilizada se basa en el modelo de implementación CRISP-DM la cual tiene por partes:

- Entendimiento del Negocio.
- Comprensión de los Datos.
- Preparación de los Datos.

- Implementación del Modelo.
- Evaluación del Modelo.
- Despliegue de Resultados.

Destacando la parte referente a la comprensión y preparación de los datos.

Para la implementación se utilizaron tres métodos en concreto:

- Regresión Logística.
- k-Vecino más Cercano (k-NN).
- Árboles de Decisión.

Cada uno siendo evaluado mediante matrices de confusión y curvas de ROC.

1.4. Contenido

1.4.1. Conceptos Básicos

Este capítulo proporciona una base sólida para que se comprendan los conceptos fundamentales de machine learning y estadística que se utilizarán a lo largo de la tesis. Además, establece la conexión entre estos dos campos, destacando la importancia de la estadística en el desarrollo y evaluación de modelos de machine learning.

1.4.2. Modelos de Regresión

Este capítulo señala conocimientos fundamentales de los modelos de regresión en el contexto del aprendizaje automático. Se exploran los principios básicos de la regresión, incluyendo diferentes tipos como la regresión lineal simple, la regresión lineal múltiple y la regresión logística, junto con sus aplicaciones y consideraciones clave en su implementación. Además, se destaca la conexión entre los modelos de regresión y la estadística, resaltando cómo los principios estadísticos, como la estimación de parámetros y la evaluación, influyen en el desempeño y la interpretación de estos modelos. Este capítulo sienta las bases para comprender en profundidad los modelos de regresión y su papel dentro del panorama más amplio del aprendizaje automático.Los temas en concreto son:

- Regresión lineal simple, múltiple y logística:

  Teoría, definiciones y aplicación práctica.

- Métodos de evaluación de rendimiento:

  Métodos para analizar y determinar la fiabilidad de los modelos.

- Ejemplo con Python:

  Ejemplo práctico en lenguaje Python con la teoría y definiciones previas en problemas reales.

1.4.3. k-Vecino más Cercano

Este capítulo trata de los conceptos fundamentales del algoritmo k-vecinos más cercanos (k-NN) en el contexto del aprendizaje automático. Se exploran los principios básicos del k-NN, incluyendo su funcionamiento, aplicaciones y consideraciones clave en su implementación. Además, se destaca la conexión entre el k-NN y la estadística, resaltando cómo los principios estadísticos,

como la distancia euclidiana y la selección del valor k, influyen en el desempeño y la evaluación de este algoritmo. Los temas en concreto son:

- Distancia:

  Sentido y definición de distancia y tres tipos de distancia: Euclidiana, Manhattan y Mahalanobis.

- Similitud:

  Definición y sentido del k-vecino más cercano.

- Ejemplo con Python:

  Ejemplo práctico en lenguaje Python con la teoría y definiciones previas en problemas reales.

1.4.4. Árboles de Decisión

Este capítulo explica la metodología de los árboles de decisión en el contexto del aprendizaje automático. Se exploran los principios básicos de los árboles de decisión, incluyendo su estructura y algoritmos de construcción, junto con su aplicación en la clasificación y regresión de datos. Además, se destaca la conexión entre los árboles de decisión y la estadística, resaltando cómo los principios estadísticos, como la entropía y el índice Gini, influyen en la construcción y evaluación de estos modelos. Este capítulo sienta las bases para comprender en profundidad los árboles de decisión y su papel dentro del panorama más amplio del aprendizaje automático. Los temas en concreto son:

- Estructura:

  Componentes y partes de un árbol de decisión.

- Método CART:

  Classification and Regression Trees: Método de árboles de decisión para la clasificación de instancias.

- Profundidad:

  Definición y sentido de la profundidad de los árboles para la implementación.

- Ejemplo con Python:

  Ejemplo práctico en lenguaje Python con la teoría y definiciones previas en problemas reales.

1.4.5. Caso de Estudio

En este capítulo se aplica la implementación de métodos de machine learning para el pronóstico dentro de un ambiente de datos de índole social de la Fundación JUCONI. Esto es especialmente relevante en entornos sociales donde las interacciones humanas y los factores contextuales pueden ser altamente variados y difíciles de modelar con enfoques convencionales.

Además, se hace notoria la flexibilidad y escalabilidad de las herramientas de machine learning que permiten adaptarse dinámica mente a cambios en los datos y en el entorno social subyacente. Esto facilita una mayor agilidad y capacidad de respuesta en la toma de decisiones basadas en evidencia.

1.4.6. Conclusiones Generales

Este capítulo denota la importancia del preprocesamiento de datos para garantizar la calidad en la implementación de modelos en el ámbito del aprendizaje automático, especialmente cuando se aplican a problemas de índole social. Se resalta que la mejora en la calidad de los datos es fundamental, ya que incluso con una implementación correcta, si los datos son deficientes, los resultados obtenidos serán insatisfactorios. Se compara la implementación de modelos con resolver un rompecabezas sin una imagen guía, destacando el desafío que implica encontrar el mejor modelo y su interpretación final. Además, se enfatiza que al aplicar modelos de aprendizaje automático a problemas sociales, no solo se amplía el campo de aplicación, sino que también se crea conciencia sobre las condiciones en las que viven las personas. Además de invitar a salir de la burbuja y conocer realmente el entorno en el que habitamos, señalando que ni el aprendizaje automático ni la inteligencia artificial pueden lograrlo, y planteando preguntas sobre la inclusión y el compromiso social en el futuro.

El conocimiento es poder, pero la comprensión de los conceptos básicos es la base de ese poder.

Hermione Granger, Harry Potter y el Cáliz de Fuego.

2 Conceptos Básicos

Este capítulo proporciona una base sólida para que se comprendan los conceptos fundamentales de machine learning y estadística para su implementación.

2.1. Big Data

Con la llegada de la *ciencia de datos* el uso de la información, como medio de conocimiento, ha dado un cambio agigantado en la última década. Tanto así que el manejo de información representada en un modelo de datos es imprescindible para la solución de problemas y el inicio de nuevos tipos de pensamiento. Es por ello que, al entorno donde habitan una gran cantidad de datos, se le suele denominar como **Big Data**. En general, big data es un término para conjuntos de datos masivos con una estructura variada, compleja y extensa [30]. Aunque se puede ahondar mucho más en este concepto, la idea de un *conjunto de datos estructurado* es lo esencial para este caso en particular.

Un **modelo de aprendizaje automático** utiliza este entorno de big data para generar información sobre la población objetivo y realizar metodologías para el planteamiento de hipótesis sobre los datos. Al implementar un modelo de aprendizaje automático a un entorno de big data se tienen que tener en cuenta ciertas metodologías establecidas por el gremio conocidas como *procesos de implementación* [2].

2.2. Procesos de Implementación

Se le conoce como *proceso de implementación* a una serie de pasos que se aplican a los datos para la gestión, consulta e implementación de modelos de minería de datos. Existen diversos procesos de implementación dependiendo del enfoque del modelo pero, de manera general, se nombran los procesos: *KDD*, *SEMMA* y *CRISP-DM*.

2.2.1. Proceso *KDD*

El *proceso de descubrimiento de conocimiento en bases de datos* (*Knowledge Discovery in Databases*), KDD por sus siglas en inglés, utiliza la minería de datos para extraer información valiosa [2]. Técnicas de preprocesamiento, muestreo y transformación a una base de datos son aplicados

según sea necesario para extraer tal información. Este proceso consta de cinco etapas descritas en la Figura 2.1:

1. Selección de Datos:
   Consiste en la creación de la base tomando en cuenta todas las variables del modelo que se desea realizar. Con esto se debe tener en cuenta el diseño de la base para operar los siguientes pasos.

2. Pre procesamiento:
   Todo lo referente a ordenar, limpiar y conocer los datos. En esta parte se considera como está constituida la base para conocer, apriori, los datos que serán transformados.

3. Transformación:
   Se transforma la data utilizando métodos para la homologación y, si es posible, la reducción de esta sin afectar al modelo posterior.

4. Minado:
   En esta parte se engloba el análisis y la aplicación de modelos, generalmente predictivos, para la búsqueda de patrones y representación de los mismos.

5. Interpretación:
   Con los datos y resultados obtenidos en todo el proceso de minado, se procede a evaluarlos y llegar a una conclusión general del modelaje.

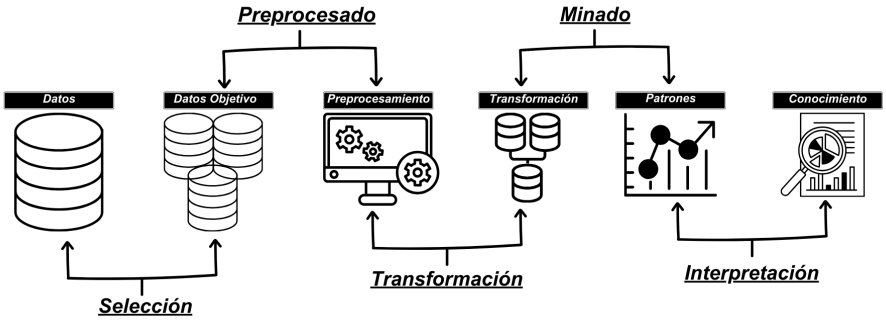

Figura 2.1: Diagrama de Implementación KDD.

2.2.2. Proceso *SEMMA*

El *proceso de muestreo, exploración, modificación, modelación y evaluación* (*Sample, Explore, Modify, Model and Assess*), SEMMA por sus siglas en inglés, se refiere a una serie de pasos para la implementación de modelos de minería de datos [2]. Consta de cinco pasos que se ejecutan de manera cíclica y se distinguen en la Figura 2.2.

1. Muestreo (***Sample***): Se extraé una muestra representativa de los datos en bruto lo bastante pequeña para una fácil manipulación. Si dichos datos no son masivos, este paso se puede obviar.

2. Exploración (***Explore***): Se explora la base a detalle para detectar posibles anomalías, redundancias o información no relevante para el análisis. Además de comprender de mejor

manera la estructura y naturaleza de los datos en bruto.

3. Modificación (***Modify***): Se modifican los datos para seleccionar, crear y transformar las variables de acuerdo a las condiciones del modelo (o modelos) para llevarlo a cabo.

4. Modelación (***Model***): Se comienza con la implementación del modelo a la base de datos para generar resultados que ayuden en la búsqueda del mejor ajuste al modelo con la mayor fiabilidad.

5. Evaluación (***Assess***): Se evalúa la calidad y utilidad de los resultados obtenidos para su presentación e interpretación.

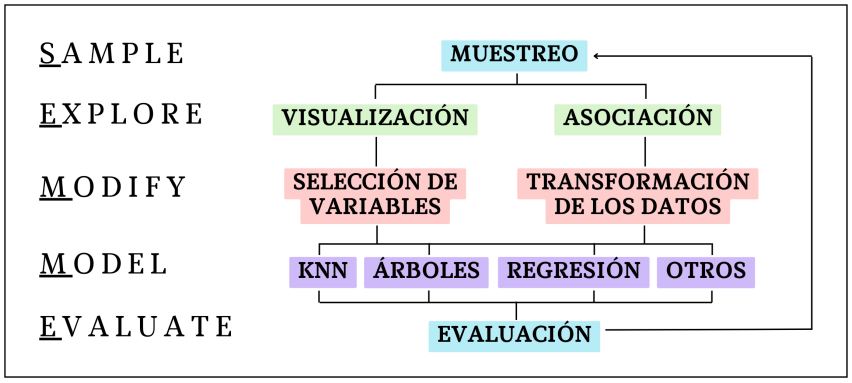

Figura 2.2: Diagrama de Implementación SEMMA.

2.2.3. Proceso: *CRISP-DM*

El *proceso estándar interprofesional para la minería de datos* (*Cross-Industry Standard Process for Data Mining*, CRISP-DM por sus siglas en inglés) se desarrolló por medio de un consorcio compuesto por DaimerChryrler, SPSS y NCR para crear una metodología estándar para la implementación de modelos de minería de datos desde el enfoque empresarial [2]. Consta de seis pasos cíclicos ilustrados en la Figura 2.3.

1. Entendimiento del Negocio: Se centra en entender los objetivos y requisitos del proyecto desde un enfoque empresarial para transformarlo en un problema de minería de datos.

   Por ejemplo, supóngase que se tiene un cliente que quiere conocer los cinco productos más vendidos en cada estado de la república. El problema en cuestión se resuelve, desde el ámbito de la ciencia de datos, con la construcción de una tabla que muestre el volumen de venta de cada producto etiquetado por estado. En este caso el objetivo es conocer la información de los cinco más vendidos pero, el enfoque en minería, es la construcción de una base de datos que otorgue dicha información.

2. Comprensión de los Datos: Se recopilan los datos para manipularlos e identificar problemas en la calidad; además de tener un primer acercamiento al comportamiento de la base y detectar subconjuntos de interés. Todo esto para generar hipótesis sobre la base en bruto.

3. Preparación de los Datos: Se transforman, modifican y crean los datos necesarios para efectuar el modelo a partir de la comprensión de los datos en bruto.

4. Implementación del Modelo: Se aplica el modelo a la base de datos y se prepara para calibrar los parámetros en búsqueda del modelo que mejor se ajuste a la naturaleza del problema.

5. Evaluación del Modelo: Se evalúan a fondo la implementación del modelo, los resultados obtenidos y la interpretación de estos para someterse a cuestionamientos sobre la utilidad

y el alcance de la solución al problema inicial.

6. Despliegue de Resultados: Se presentan los resultados de manera que se resuelva la cuestión inicial de manera sencilla.

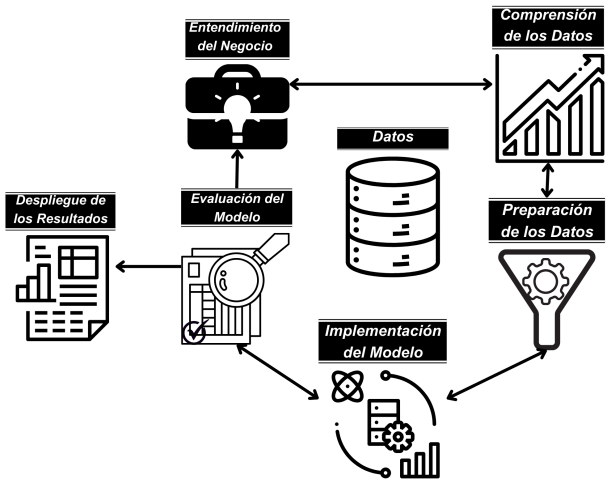

Figura 2.3: Diagrama de implementación CRISP-DM.

2.3. Fundamentos de Bases de Datos

Todo el tema que engloba al conocimiento teórico de *bases de datos*, desde el método hasta su visualización, es en efecto, bastante amplia. Al hablar de sus fundamentos se pueden mencionar el diseño, el almacenamiento, la gestión, la arquitectura, las relaciones entre bases y la visualización de las mismas, entre otras. En particular se busca que la base de datos sea homogénea y estructurada (o diseñada) de tal manera que se realicen métodos y operaciones con la misma.

Cuando se habla del diseño se busca una estructura lógico para la construcción de un modelo de base de datos. Para esto existen varios tipos de modelos utilizados en el diseño de bases de datos. Como los son: relacional, entidad-relación, basado en objetos o semiestructurado entre otros.

Diseño de Bases de Datos

Modelo Entidad-Relación

Se suele usar el modelo *entidad-relación* (E-R); el cual consta de una asociación entre bases de datos (llamadas *entidades*) y el cómo se relacionan entre sí[34].

Dentro del modelo entidad-relación las entidades se definen como grupos de atributos de la base de datos principal etiquetados de manera puntual para cada entrada con algún identificador propio único. Una relación es la asociación entre entidades mediante algún identificador [3].

Al grupo de entidades del mismo tipo y al grupo de relaciones del mismo tipo se les conoce como *conjunto de entidades* y *conjunto de relaciones* respectivamente cómo se ilustra en la Figura 2.4.

Se representar el modelo E-R mediante un diagrama con los siguientes componentes:

- Rectángulo: Representa a los conjuntos de entidades.
- Elipse: Representa a los atributos.
- Rombo: Relaciones entre conjuntos de entidades.

- Línea: Unión de los atributos con las entidades y éstas a su vez con las relaciones.

Figura 2.4: Diagrama de modelo de entidad-relación.

2.4. Fundamentos Estadísticos

Regresión

Se dice que se tiene un problema de regresión cuando el objetivo es predecir un resultado del tipo numérico; además de que estos problemas pertenecen a un entorno continuo. La Figura 2.5 muestra una gráfica de cómo crece el salario con base en los años de experiencia (puntos de colo rojo) y la predicción que se obtuvo con una regresión lineal simple (recta de color azul) aplicada al conjunto de entrenamiento y, a la derecha, con el conjunto de prueba[28].

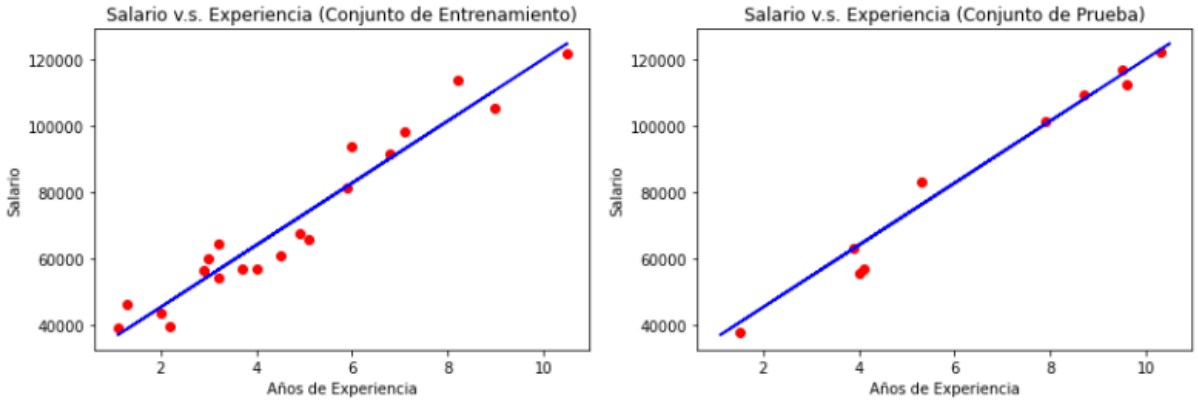

Figura 2.5: Ejemplo de una regresión lineal simple para predecir el salario a partir de los años de experiencia; tanto con el conjunto de entrenamiento como el conjunto de prueba.

Clasificación

Se tiene un problema de clasificación cuando el objetivo es predecir un resultado del tipo categórico; además de que estos problemas pertenecen a un entorno discreto. Ejemplos de clasificación son tales como:

- Conocer si un cliente canjeó o no un cupón.
- Opiniones acerca de algún tema en particular.
- Tipos de violencia que se generan en un entorno determinado.

La Figura 2.6 muestra dos ejemplos de clasificación; un ejemplo general de un árbol de decisión y una clasificación por grupos. Es muy frecuente que, dentro de los problemas de clasificación, se opte por predecir probabilidades de clases, es decir, la probabilidad de que alguna característica ocurra[5].

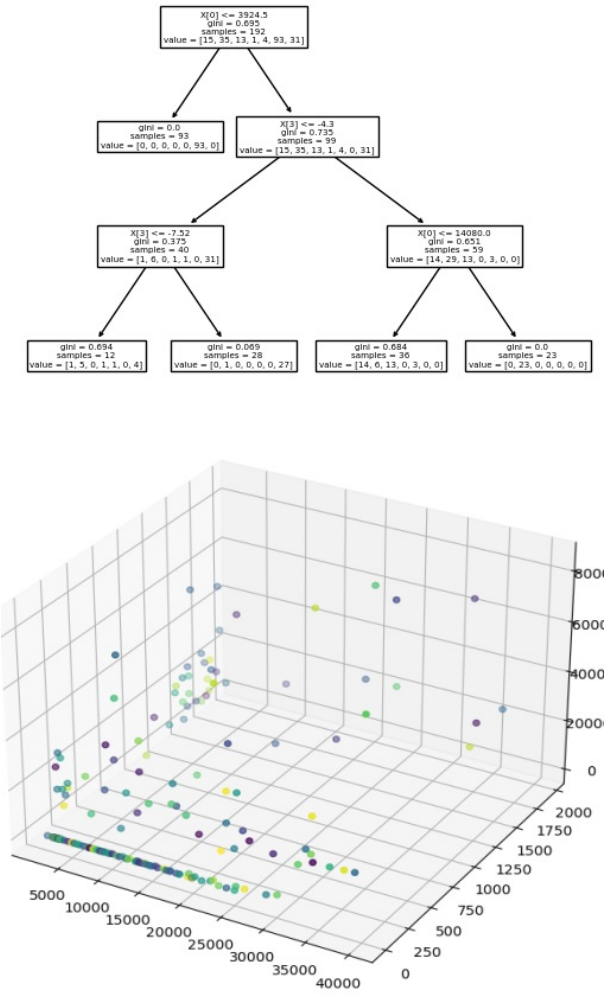

Figura 2.6: Ejemplo de dos métodos de clasificación. En la parte superior se muestra una representación de un árbol de decisión y, en la parte inferior, una agrupación relacionada con el árbol.

Algunas medidas de clasificación utilizadas en árboles de decisión son el *índice Gini* y la *Entropía* las cuales se encargan de medir la pureza de los nodos para realizar la clasificación. En el capítulo dedicado a árboles de decisión se ahonda más en éstos métodos y la diferencia a la hora de aplicarlos.

2.5. Ciencia de Datos

2.5.1. Inteligencia Artificial

La inteligencia artificial, -fuera de la ciencia ficción-, es una rama de la computación que tiene por objetivo que los ordenadores hagan la misma clase de tareas que puede realizar la mente humana dentro de un entorno estructurado. De aquí nace una rama llamada *aprendizaje automático* (*machine learning* en inglés) la cual se encarga de que dichos ordenadores sigan una metodología de aprendizaje a partir de ordenes dadas por medio de código para ejecutar un propósito en concreto o una variedad de tareas[4].

2.5.2. Aprendizaje Automático

El *aprendizaje*, como tal, hace referencia a las situaciones en que un *aprendiz* incrementa sus conocimientos o habilidades para cumplir un propósito. Este aprendiz debe ser capaz de tomar decisiones sobre el curso que debe seguir la resolución del problema. El aprendizaje automático toma los conceptos para aplicar al ámbito computacional, a modo de enseñarle a una máquina como resolver problemáticas a partir de lo aprendido. A este proceso de *aprendizaje* se le conoce como *entrenamiento* [18].

Dentro del aprendizaje automático diferenciamos entre dos tipos: el aprendizaje supervisado y el aprendizaje no supervisado.

Aprendizaje Supervisado

En el aprendizaje supervisado se asumen datos etiquetados; es decir, con una variable de respuesta y un conjunto de variables predictoras (también llamadas variable dependiente y variables independientes). La meta dentro de este tipo de aprendizaje es explicar la variable respuesta en términos de las variables predictoras y así crear modelos predictivos. Por ejemplo:

- Predecir precios de tiendas de conveniencia a partir de los datos (o atributos) de los consumidores.
- Predecir la deserción o rotación de empleados.
- Predecir el riesgo de que un paciente, al darlo de alta, regrese al hospital después de determinado número de días.

Dentro del aprendizaje supervisado podemos encontrar los métodos de regresión y los métodos de clasificación donde el algoritmo de aprendizaje de cada uno intenta optimizar la función objetivo[1] para hallar la mejor combinación de valores para que dicha función objetivo se ajuste lo mejor a la realidad[35].

Aprendizaje No Supervisado

En el aprendizaje no supervisado los datos son no etiquetados e incluye herramientas estadísticas para describir mejor los datos sin la necesidad de una variable o función objetivo. Este tipo de aprendizaje se ocupa de identificar grupos en un conjunto de datos ya sea para la agrupación en conglomerados de las observaciones similares en función de las variables que se observan o reducir las dimensiones disminuyendo el número de variables en el conjunto de datos correspondiente. Sin embargo hay que tener en cuenta que este tipo de aprendizaje tiende a la subjetividad ya que, a diferencia del aprendizaje supervisado, no es posible verificar que tan bien se ajusta a la realidad el modelo por falta de indicadores que denoten la precisión y veracidad de los resultados predecidos. Es por eso mismo que se le llama *no supervisado*. Pero, a pesar de tener esta tendencia, el aprendizaje no supervisado es muy útil para[35]:

- Dividir consumidores para aplicar técnicas de marketing.
- Identificar individuos que cumplan ciertas características similares para tener una visualización más clara de los datos.
- Identificar comportamientos para el posterior descubrimiento de patrones.
- Realizar un análisis exploratorio previo de los datos.

[1] Función objetivo: Es la ecuación que será optimizada dadas las limitaciones o restricciones determinadas.

En posteriores capítulos se desarrollarán estos conceptos con su respectiva metodología y su aplicación final en el caso de estudio con una base de datos de la Fundación JUCONI A. C. con características de jóvenes que participaron en programas de la Fundación.

No hay que ser un Nostradamus para saber a dónde vamos,
Ni un profeta para predecir el mañana,
Solo abre los ojos y tendrás premoniciones,
Tú, yo, cualquiera podría tener visiones.

Canserbero, Visiones.

3

Modelos de Regresión

Desde tiempos inmemorables se ha hablado sobre lo que se conoce como *predicción* en diferentes culturas. Los oráculos en Grecia, los profetas en cada una de sus épocas entre otros.

Fuera de la fantasía, y la especulación sesgada, el planteamiento de un modelo estadístico que permita *ajustarse* de la mejor manera a la realidad se le conoce como *modelo de predicción o predictivo*. Los modelos de regresión son parte de estos los cuales utilizan metodologías y planteamientos estocásticos para conocer el comportamiento de la población y hacer inferencia sobre los datos. Utilizando como variable dependiente (u objetivo) al valor que se desea predecir desde el conocimiento de una o más variables independientes que estén relacionados, por ejemplo, los siguientes tres tipos de regresión: simple, múltiple y logística [32].

Dentro del ámbito de aprendizaje automático, aunque se les considera clásicos, es común la aplicación de modelos de regresión dentro del aprendizaje supervisados.

3.1. Regresión Lineal Simple

Se le denomina *simple* debido a que solo contiene *una variable dependiente* en la ecuación. Este modelo es de la forma:

$$Y = \beta_0 + \beta_1 x + \varepsilon. \tag{3.1}$$

En donde:

- $Y :=$ Variable dependiente, también conocida como variable respuesta la cual es una *variable aleatoria observable*.
- $x :=$ Variable independiente, también conocida como variable predictora, considerada como no aleatoria o bien como una *observación de una variable aleatoria*.
- β_0 y β_1 son los parámetros a estimar del modelo. Donde:
  - β_0 es el intercepto, es decir, el valor de y cuando $x = 0$. En un contexto práctico, representa el valor estimado de la variable dependiente cuando todas las variables independientes son iguales a cero.

- β_1 es la pendiente de la línea de regresión, es decir, el cambio en y por cada unidad de cambio en x.Este coeficiente indica la relación de cambio entre la variable independiente x y la variable dependiente y.

- $\varepsilon :=$ El error de la estimación, la cual es una variable aleatoria no observable.

Esta ecuación, vista desde la perspectiva del aprendizaje automático, se interpreta como un modelo del cual solo se conocen dos características: la variable independiente y la variable dependiente [5]. Pero, en la práctica, se tiene un número determinado de observaciones de las características; dichas observaciones están dadas por las entradas de la base de datos. Con esto se reescribe la Ecuación (3.1) de la siguiente forma:

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \;\; i = 1, ..., n. \tag{3.2}$$

En donde:

- $Y_i :=$ i-ésima observación de la variable dependiente.
- $x_i :=$ i-ésima observación de la variable independiente.
- β_0 y β_1 son los parámetros a estimar.
- $\varepsilon_i :=$ i-ésimo error de la estimación, el cual no es observable.

Haciendo uso del álgebra matricial se reescriben las ecuaciones dadas en (3.2) como:

$$\boldsymbol{Y} = \mathbb{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}. \tag{3.3}$$

donde:

- $\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}$: Vector de observaciones de la variable dependiente.
- $\mathbb{X} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$: Matriz de valores de variables dependientes donde $x_{i1} = 1$ y $x_{i2} = x_i$, para $i = 0, \ldots, n$.
- $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$: Vector de parámetros a estimar.
- $\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}$: Vector que contiene los errores del modelo de regresión.

3.1.1. Estimación

Se busca estimar a β_0 y β_1 para obtener el mejor ajuste del modelo de regresión lineal simple. Como de manera usual, se supone que, los errores $\varepsilon_i \sim \eta\left(0, \sigma^2\right)$; se plantea obtener un estimador de *máxima* verosimilitud para los parámetros partiendo de la definición de *función de verosimilitud* y de *estimador de máxima verosimilitud* [6] [28].

Definición 1 (Función de Verosimilitud)

Sea $Y_1, \ldots, Y_n$ una muestra aleatoria, es decir, un conjunto de n variables aleatorias independientes cada una con la misma distribución (iid), con función de densidad $f(y_1, \ldots, y_n; \theta_1, \ldots, \theta_k)$ la función de verosimilitud está dada por:

$$\mathbf{L}(\theta_1, \ldots, \theta_k; y_1, \ldots, y_n) = \prod_{i=1}^{n} f(y_i; \theta_1, \ldots, \theta_k) \tag{3.4}$$

considerando a los parámetros como variables y a la realización de la muestra fija.

Definición 2 (Estimador de Máxima Verosimilitud)

Sea S un conjunto de puntos muestrales, para cada punto de la muestra $y \in \mathcal{S}$, $\hat{\theta}(y)$ es una estimación donde la función de verosimilitud $\mathbf{L}(\theta_1, \ldots, \theta_k; y_1, \ldots, y_n)$ alcanza su máximo en función de θ con el vector $\boldsymbol{y}$ cuyos elementos son cantidades fijas y corresponden a una realización del vector aleatorio $\mathbf{Y}$. Un estimador de máxima verosimilitud del parámetro θ, basado en la muestra S se denotará por $\hat{\Theta}(\boldsymbol{y})$.

Estas dos definiciones abren camino para obtener estimadores de máxima verosimilitud para β_0, β_1 y σ^2 de la siguiente manera:

Estimador de Máxima Verosimilitud para el Modelo de Regresión Lineal Simple

Sea $Y = \beta_0 + \beta_1 x + \varepsilon$ el modelo de la regresión lineal simple con $\varepsilon \sim \eta\left(0, \sigma^2\right)$, de donde, resulta claro que la variable aleatoria $Y \sim \eta\left(\beta_0 + \beta_1 x, \sigma^2\right)$. De modo que la función de densidad está dada por:

$$f(y \mid x; \beta_0, \beta_1, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{y-(\beta_0+\beta_1 x)}{\sigma}\right)^2}.$$

Así, la función de verosimilitud asociada a $(\mathbf{x} \mid \mathbf{y})$, es:

$$\mathbf{L}\left(\beta_0, \beta_1, \sigma^2; y, x\right) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{n}{2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \beta_1 x_i\right)^2\right\}.$$

Es común que dentro del procedimiento para encontrar estimadores por el método de máxima verosimilitud se opte por calcular el *logaritmo natural* de la función de verosimilitud. Dicha transformación de la función de verosimilitud se denota por $l\left(\beta_0, \beta_1, \sigma^2\right)$. De modo que, calculando el logaritmo natural de la función de verosimilitud se obtiene:

$$l\left(\beta_0, \beta_1, \sigma^2\right) = -\frac{n}{2}(\ln(2\pi) - \ln\left(\sigma^2\right)) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \beta_1 x_i\right)^2.$$

Haciendo uso del criterio de la segunda derivada para obtener el punto donde se maximiza $l\left(\beta_0, \beta_1, \sigma^2\right)$, derivando con respecto a cada componente de su argumento e igualando a cero estas derivadas, se obtienen los puntos críticos y después se prueba si estos producen un máximo usando la matriz de las segundas derivadas. Se sabe que en el punto obtenido la función de verosimilitud alcanza su máximo. Así, la solución corresponde a los estimadores de máxima verosimilitud para los parámetros β_0, β_1 y σ^2, dada la realización $(y_1, x_1), ..., (y_n, x_n)$:

- $\hat{\beta}_0 = \dfrac{1}{n}\sum_{i=1}^{n} y_i - \hat{\beta}_1 \dfrac{1}{n}\sum_{i=1}^{n} x_i = \bar{y} - \hat{\beta}_1 \bar{x}.$

- $\hat{\beta}_1 = \dfrac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2} = \dfrac{s_{xy}}{s_{xx}}.$

- $\hat{\sigma}^2 = \dfrac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right)^2.$

Para asegurar que se tratan de estimaciones de máxima verosimilitud se hace uso del criterio de la segunda derivada. De modo que, calculando la segunda derivada de $l\left(\beta_0, \beta_1, \sigma^2\right)$ con respecto a cada componente de su argumento y evaluando en los valores del punto obtenido en la solución de las ecuaciones de verosimilitud se obtiene la matriz *Hessiana* y, por comodidad de notación, se denomina a $R = \{1, \ldots, n\}$ el cual es el recorrido del índice de la suma. Con esto se tiene que:

$$H = \begin{pmatrix} \frac{\partial^2 l}{\partial \beta_0^2} & \frac{\partial^2 l}{\partial \beta_1 \partial \beta_0} & \frac{\partial^2 l}{\partial \sigma^2 \partial \beta_0} \\ \frac{\partial^2 l}{\partial \beta_0 \partial \beta_1} & \frac{\partial^2 l}{\partial \beta_1^2} & \frac{\partial^2 l}{\partial \sigma^2 \partial \beta_1} \\ \frac{\partial^2 l}{\partial \sigma^2 \partial \beta_0} & \frac{\partial^2 l}{\partial \sigma^2 \partial \beta_1} & \frac{\partial^2 l}{\partial (\sigma^2)^2} \end{pmatrix}_{\left[\hat{\beta}_0, \hat{\beta}_1, \hat{\sigma}^2\right]} =$$

$$\begin{pmatrix} \frac{-1}{\sum_{i \in R}\left(\bar{x}y - \bar{y} + \frac{S_{x,y}}{S_{xx}}(x_i - \bar{x})\right)^2} & \frac{-1}{\sum_{i \in R}\left(\bar{x}y - \bar{y} + \frac{S_{x,y}}{S_{xx}}(x_i - \bar{x})\right)^2} & 0 \\ \frac{-n\sum_{i \in R} x_i}{\sum_{i \in R}\left(y - \bar{y} + \frac{S_{x,y}}{S_{xx}}(\bar{x} - x_i)\right)} & -\frac{1}{\sigma^2}\sum_{i \in R} x_i^2 & \frac{1}{\sum_{i \in R} y_i - \bar{y} + \frac{S_{xy}}{S_{xx}}\bar{x} - \frac{S_{xy}}{S_{xx}}x_i} \\ \frac{1}{\sum_{i \in R} y_i - \bar{y} + \frac{S_{xy}}{S_{xx}}\bar{x} - \frac{S_{xy}}{S_{xx}}x_i} & \frac{-n}{\sum_{i \in R} y_1 - \hat{\beta}_0 - \hat{\beta}_i x_i} & \frac{n^2}{2\sum_{i \in R}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2} - \frac{n^3}{2\sum_{i \in R}(y_i - \hat{\beta}_0 - \hat{\beta}_1)x_i} \end{pmatrix}.$$

Esta matriz es definida negativa, por lo que se puede asegurar que los estimaciones para las β's y para σ^2 son, en efecto, de máxima verosimilitud. Debido a que estas estimaciones tienen la misma forma para cualesquiera realización del vector $\mathbf{Y}$, entonces los estimadores de máxima verosimilitud tendrán la misma forma, solo cambiando las realizaciones por sus correspondientes variables aleatorias. En lo sucesivo se usará una sola notación (Y, y) para representar una variable aleatoria o su realización.

Aunque existe literatura la cual dice que, para probar que los estimadores calculados son de máxima verosimilitud, solo basta con calcular las derivadas de la diagonal principal de la matriz H y asegurarse de que sean menor a cero; es mejor calcular la matriz y conocer como se define para asegurar la verosimilitud de los estimadores calculados.

En cuánto al cálculo de estimadores, se encuentran también las definiciones de lo que es el *error cuadrado medio* y los estimadores *insesgados* [6].

Definición 3 (Error Cuadrático Medio)

El error cuadrático medio (ECM) de un estimador $\hat{\theta}$ para un parámetro θ es una función de θ definida por:

$$\mathrm{E}[\hat{\theta} - \theta]^2 = Var\left(\hat{\theta}\right) + \left(\mathrm{E}\left[\hat{\theta}\right] - \theta\right)^2. \tag{3.5}$$

La diferencia que se encuentra en el último sumando de la ecuación se conoce como ***sesgo*** *[6] [28].*

Definición 4 (Mejor Estimador Insesgado)

Un estimador $\hat{\theta}$ es el mejor estimador insesgado para θ si satisface que $\mathrm{E}[\hat{\theta}] = \theta$ y, para cualquier otro estimador $\hat{\theta}^*$ con $\mathrm{E}[\hat{\theta}^*] = \theta$, se tiene que $Var[\hat{\theta}] \leq Var[\hat{\theta}^*]$. $\hat{\theta}$ es también conocido como el ***estimador insesgado de varianza uniformemente mínima*** *para θ.*

Definición 5 (Mínimos Cuadrados)

Sea $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ n-pares de números trazados en una gráfica de dispersión. Supóngase que existe una recta que pasa a través del grupo de puntos que se acerca lo más posible a todos los puntos. Para cualquier recta, la suma de cuadrados residual (SCR) se define como:

$$SCR = \sum_{i=1}^{n} (by_i - \beta_0 - \beta_1 x_i)^2 . \tag{3.6}$$

Mínimos Cuadrados y Estimador insesgado de varianza uniformemente mínima

El cálculo de estimadores para β_0 y β_1 mediante mínimos cuadrados es como sigue:

Sea $\zeta(\beta_0, \beta_1) = \sum_{i=1}^{n} \varepsilon_i^2$, tomando como $\varepsilon = y_i - \beta_0 - \beta_1 x_i$ se deriva dicha función ζ con respecto a cada componente de su argumento y se iguala a cero de la siguiente manera:

- $\frac{\partial}{\partial \beta_0}\zeta(\beta_0, \beta_1) = -2(\sum_{i=1}^{n} y_i - \beta_0 - \beta_1 x_i) = 0.$
- $\frac{\partial}{\partial \beta_1}\zeta(\beta_0, \beta_1) = \sum_{i=1}^{n} 2(y_i - \beta_0 - \beta_1 x s_i)(-x_i) = 0.$

Se puede observar que estas ecuaciones son las mismas que se han obtenido como ecuaciones de verosimilitud, por lo que en este caso, los estimadores para β_0 y β_1 son:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}.$$

y

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{s_{xy}}{s_{xx}}.$$

Para la parte del **mejor estimador insesgado de varianza uniformemente mínima** usualmente se empieza por calcular la esperanza de cada estimador y verificar si la definición se cumple. Esto es cierto ya que los estimadores calculados con anterioridad son insesgados.

Definición 6 (Estimador Insesgado) *Un estimador $\hat{\theta}$ del parámetro θ es insesgado si se cumple que[26]:*

$$\mathbb{E}[\hat{\theta}] = \theta.$$

Se procede entonces a probar que $\hat{\beta}_0$ y $\hat{\beta}_1$ son estimadores insesgados para los parámetros β_0 y β_1.

Prueba

Sean $\hat{\beta}_0, \hat{\beta}_1$ estimadores de los parámetros β_0, β_1 se busca probar que dichos estimadores son insesgados.

•Para $\hat{\beta}_1$ se sabe que:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}.$$

Lo cual se reescribe denotando a $\mathcal{X}_i = x_i - \bar{x}$, de este modo se reformula a $\hat{\beta}_1$ como:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{\sum_{i=1}^{n}\mathcal{X}_i(y_i - \bar{y})}{\sum_{i=1}^{n}\mathcal{X}_i^2} = \frac{\sum_{i=1}^{n}[\mathcal{X}_i y_i - \mathcal{X}_i \bar{y}]}{\sum_{i=1}^{n}\mathcal{X}_i^2}.$$

Como la suma es un operador lineal, se pueden separar los sumandos y extraer las constantes. Dando entonces lo siguiente:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} \mathcal{X}_i y_i - \bar{y} \sum_{i=1}^{n} \mathcal{X}_i}{\sum_{i=1}^{n} \mathcal{X}_i^2} = \frac{\sum_{i=1}^{n} \mathcal{X}_i y_i}{\sum_{i=1}^{n} \mathcal{X}_i^2}.$$

Esto ocurre ya que el término $\bar{y} \sum_{i=1}^{n} \mathcal{X}_i = 0$ lo cual sucede por lo siguiente: $\sum_{i=1}^{n} \mathcal{X}_i = \sum_{i=1}^{n} (x_i - \bar{x}) = \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \bar{x} = n\bar{x} - n\bar{x} = 0$.

Se denota a $T_i = \frac{\sum_{i=1}^{n} \mathcal{X}_i}{\sum_{i=1}^{n} \mathcal{X}_i^2}$ para reescribir a $\hat{\beta}_1$ finalmente como:

$$\hat{\beta}_1 = \sum_{i=1}^{n} T_i y_i.$$

Desarrollando la ecuación anterior con $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ se obtiene lo siguiente:

$$\hat{\beta}_1 = \sum_{i=1}^{n} T_i y_i = \sum_{i=1}^{n} T_i(\beta_0 + \beta_1 x_i + \epsilon_i) = \beta_0[\sum_{i=1}^{n} T_i] + \beta_1[\sum_{i=1}^{n} T_i]x_i + [\sum_{i=1}^{n} T_i]\epsilon_i.$$

Lo cual se reescribe como:

$$\hat{\beta}_1 = \beta_1 + [\sum_{i=1}^{n} T_i]\epsilon_i.$$

Aplicando el valor esperado a ambos lados de la igualdad:

$$\mathbb{E}\left[\hat{\beta}_1\right] = \mathbb{E}\left[\beta_1 + [\sum_{i=1}^{n} T_i]\epsilon_i\right] = \mathbb{E}\left[\beta_1\right] + \mathbb{E}\left[[\sum_{i=1}^{n} T_i]\epsilon_i\right] = \beta_1 + \sum_{i=1}^{n} T_i \mathbb{E}\left[\epsilon_i\right] = \beta_1.$$

Entonces:

$$\mathbb{E}\left[\hat{\beta}_1\right] = \beta_1.$$

Por lo tanto $\hat{\beta}_1$ es un estimador insesgado.

• Para β_0 se tiene que:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} = \frac{\sum_{i=1}^{n} y_i}{n} - \hat{\beta}_1 \bar{x}.$$

Aplicando el valor esperado a ambos lados de la igualdad:

$$\mathbb{E}\left[\hat{\beta}_0\right] = \mathbb{E}\left[\frac{\sum_{i=1}^{n} y_i}{n} - \hat{\beta}_1 \bar{x}\right] = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[y_i] - \mathbb{E}[\hat{\beta}_1 \bar{x}].$$

Como $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ se puede reescribir la ecuación de la siguiente forma:

$$\mathbb{E}\left[\hat{\beta}_0\right] = \frac{\sum_{i=1}^{n} \mathbb{E}\left[\beta_0 + \beta_1 x_i + \epsilon_i\right]}{n} - \mathbb{E}[\hat{\beta}_1 \bar{x}] = \frac{\sum_{i=1}^{n} \beta_0 + \beta_1 x_i}{n} - \hat{\beta}_1 \bar{x} =$$

$$= \frac{\sum_{i=1}^{n} \beta_0}{n} + \frac{\beta_1 \sum_{i=1}^{n} x_i}{n} - \beta_1 \bar{x}.$$

Desarrollando la ecuación se obtiene lo siguiente:

$$\mathbb{E}\left[\hat{\beta}_0\right] = \frac{n\beta_0}{n} + \frac{\beta_1 n \bar{x}}{n} - \beta_1 \bar{x} = \beta_0.$$

Entonces:

$$\mathbb{E}\left[\hat{\beta}_0\right] = \beta_0.$$

Con esto queda probado que, tanto $\hat{\beta}_0$ como $\hat{\beta}_1$, son estimadores insesgados □.

Así como existe la definición de un estimador insesgado, de igual manera existe la definición del *sesgo*.

Definición 7 (Sesgo) *Se define como sesgo de un estimador $\hat{\theta}$ de un parámetro θ como:*

$$B(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta.$$

Con esto se puede relacionar al ECM con el sesgo mediante dos ecuaciones:

Proposición 1 (Relación entre ECM y Sesgo) *Sea $\hat{\theta}$ un estimador del parámetro θ, los siguientes enunciados se cumplen:*

- $ECM(\hat{\theta}) = Var(\hat{\theta}) + B^2(\hat{\theta})$.
- $B^2(\hat{\theta}) \leq ECM(\hat{\theta})$.

Es claro que, si el estimador $\hat{\theta}$ es insesgado el $ECM(\hat{\theta}) = Var(\hat{\theta})$. Si se da el caso de que exista otro estimador insesgado, para encontrar el de mínima varianza se recurre a la *Cota Inferior de Cramér-Rao*. Este teorema proporciona, bajo ciertas condiciones, una cota inferior para las varianzas de cualquier estimador insesgado. Pero, antes de enunciarlo, se requiere definir y enunciar algunos elementos estadísticos.

Definición 8 (Función Parametral) *Sea θ un parámetro (o vector de parámetros) de una distribución de probabilidad. Cualquier función τ que cumpla $\tau : \theta \longmapsto \tau(\theta)$ se le denomina como* ***función parametral****[26].*

Como una función parametral depende del parámetro asociado, de igual menera se define el *insesgamiento*.

Definición 9 (Función Parametral Insesgada) *Sea θ un parámetro (o vector de parámetros) y sea τ una función parametral. Una estadística T es un* ***estimador insesgado*** *para $\tau(\theta)$ si:*

$$\mathbb{E}[T] = \tau(\theta).$$

Teorema 1 (Cota Inferior de Cramér-Rao) *Sea $X_1, \ldots, X_n$ una muestra aleatoria de una distribución con función de probabilidad (o de distribución) $f(x;\theta)$, dependiente de un parámetro desconocido θ. Sea T un estimador insesgado para una función parametral τ. Se cumple que[26]:*

$$Var(T) \geq \frac{(\tau'(\theta))^2}{n\mathbb{E}[(\frac{\partial}{\partial\theta}\ln f(X,\theta))^2]}. \tag{3.7}$$

Siempre y cuando se cumplan las siguientes ***condiciones de regularidad****:*

1. *El soporte de $f(x,\theta)$ dado por el conjunto $\{x : f(x,\theta) > 0\}$ no depende de θ.*
2. *Para todo x en el soporte de $f(x,\theta)$, existe la siguiente derivada:*

$$\frac{\partial}{\partial\theta}\ln f(x,\theta).$$

3. *La siguiente conmutación es válida:*

$$\frac{\partial}{\partial\theta}\int_{\mathbb{R}} f(x,\theta)dx = \int_{\mathbb{R}}\frac{\partial}{\partial\theta}f(x,\theta)dx = 0.$$

4. $0 < \mathbb{E}[(\frac{\partial}{\partial\theta}\ln f(X,\theta))^2] < \infty$.

5. *Denotando a* x^n *como* $x^n = (x_1, \ldots, x_n)$ *la siguiente conmutación es válida:*

$$\frac{\partial}{\partial\theta}\int_{\mathbb{R}^n} T(x^n)f(x^n,\theta)dx^n = \int_{\mathbb{R}^n} T(x^n)\frac{\partial}{\partial\theta}f(x^n,\theta)dx^n.$$

Posteriormente se calculan las varianzas de los estimadores obtenidos en los modelos para su comparación y así obtener el estimador de mínima varianza.

Al calcular los estimadores de máxima verosimilitud con los de mínimos cuadrados es de notar que coinciden; es decir, ambos estimadores son los mismos. Esto implica que, para las estimaciones de las betas, cualquiera de las dos metodologías se pueden utilizar; pero, para el cálculo de estimadores para la variable aleatoria Y del modelo de regresión lineal simple, como conocemos la distribución de dichas variables es mejor utilizar máxima verosimilitud ya que se obtiene, además de las estimaciones de las betas, un estimador para la varianza [5].

Ahora conociendo esta manera de estimar el valor de la variable dependiente, el modelo de regresión estimado está dado por:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x. \tag{3.8}$$

donde:

- $\hat{y}$:= Estimación de la variable dependiente.
- x := Valor de la variable independiente.
- $\hat{\beta}_0$ y $\hat{\beta}_1$ son las estimaciones de los parámetros.

3.2. Regresión Lineal Múltiple

Este modelo de regresión se denomina como *múltiple* ya que el número mínimo de variables independientes en el modelo tiene que ser mayor o igual a dos; de igual forma se cuenta con una variable dependiente y de un número determinado de parámetros β's [28]. Quedando la ecuación de la siguiente manera:

$$Y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \varepsilon,\ k > 0 \text{ ó } k \geq 2. \tag{3.9}$$

donde:

- Y := Variable dependiente.
- x_i := Variables independientes, $i = 1, \ldots, k$.
- β_i son los parámetros a estimar, $i = 0, \ldots, k$.
- ε := Error aleatorio no observable.

Se expresa a la regresión lineal múltiple muestral, -desde el enfoque de aprendizaje automático, considerando k observaciones, representadas cada una de ellas de la forma $(y_i, x_{i0}, ..., x_{ik})$, para $i = 1, ..., k$ de la siguiente manera:

$$Y_i = \beta_0 x_{i0} + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + \varepsilon_i,\ \ i = 0, \ldots, k. \tag{3.10}$$

donde:

- Y_i := Valor de la variable dependiente en la i-ésima observación..
- x_{ij} := Variable independiente i; también conocida como la i-ésima observación de la j-ésima variable dependiente $i = 1, \ldots, k$. $j = 0, \ldots, n$.

- β_i son los parámetros a estimar, $i = 0, \ldots, n$.
- $\varepsilon_i :=$ Error de la estimación (valor de la variable aleatoria no observable).

Usualmente se supone que la primera variable independiente $x_{j0} = 1$ para todo i. Transformando la Ecuación (3.9) en la siguiente:

$$Y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + \varepsilon_i; \;\; i = 1, \ldots, n. \tag{3.11}$$

y a β_0 en particular se le conoce como *intercepto de la regresión.*

De aquí se puede hacer uso de la forma matricial de la regresión lineal múltiple como sigue:

$$\mathbf{Y} = \mathbb{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}. \tag{3.12}$$

En cuanto a la representación, es de notar que la Ecuación (3.12) se representa igual que la Ecuación (3.3); el cambio aquí es dentro de la matriz de los valores de las variables independientes $\mathbb{X}$ (Esta matriz, algunos autores le llaman matriz diseño). Escrita de forma matricial la Ecuación (3.12) , se tiene lo siguiente:

- $\mathbf{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_k \end{bmatrix} :=$ Es el vector de los valores de las variables dependientes.
- $\mathbb{X} = \begin{bmatrix} 1 & x_{11} & \ldots & x_{k1} \\ 1 & x_{12} & \ldots & x_{k2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & \ldots & x_{kn} \end{bmatrix} :=$ Es la matriz de valores de variables independientes.
- $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_n \end{bmatrix} :=$ Es el vector de parámetros a estimar.
- $\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_k \end{bmatrix} :=$ El vector aleatorio que contiene los errores de estimación.

Existen, además, una serie de supuestos dentro de la regresión múltiple. Estos supuestos generalmente se infringen mientras se incluyan una mayor cantidad de variables predictoras x al modelo. Dicha infracción lleva a una mala interpretación de los coeficientes de regresión y de la predicción obtenida en si misma [37]. Dichos supuestos son:

1. **Relación lineal**. En cualquier combinación de valores $x_1, \ldots, x_n$, la relación con la media de Y es lineal.
2. **Homocedasticidad**. En cualquier combinación de valores $x_1, \ldots, x_n$, a diferentes combinaciones, la varianza del error, se mantiene constante.
3. **Independencia**. Cualquier valor del error es independiente a cualquier otro valor del error.
4. **Normalidad**. Para cualquier combinación de valores de $x_1, \ldots, x_n$, los errores tienen una distribución normal.

3.2.1. Estimación

Se puede entender al modelo de regresión lineal múltiple como una extensión del modelo de regresión lineal simple, por lo que es una opción utilizar la metodología de *mínimos cuadrados*. De modo que la estimación puntual de mínimos cuadrados se calcula de la forma:

$$\hat{\boldsymbol{\beta}} = \begin{pmatrix} \hat{\beta}_0 \\ \vdots \\ \hat{\beta}_k \end{pmatrix} = (\mathbb{X}^\tau \mathbb{X})^{-1} \mathbb{X}^\tau \mathbf{Y}.$$

En donde el vector columna $\mathbf{Y}$ y la matriz diseño $\mathbb{X}$ se utilizan para calcular las estimaciones. Ahora, sean $\hat{\beta}_0, \ldots, \hat{\beta}_n$ las estimaciones puntuales de mínimos cuadrados de los parámetros $\beta_0, \ldots, \beta_n$ del modelo de regresión lineal y suponer además que $\hat{x}_1, \ldots, \hat{x}_n$ son los valores especificados de las variables aleatorias independientes $x_1, \ldots, x_n$ la ecuación:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \hat{x}_1 + \cdots + \hat{\beta}_n \hat{x}_n \tag{3.13}$$

es la estimación puntual del valor medio de la variable dependiente cuando los valores de las variables independientes son: $x_{01}, \ldots, x_{0k}$.

Variables Dummy

Dentro de la práctica muchas veces se presentan bases de datos con categorías no numéricas. Es en estos casos donde se hace uso de las variables *dummy*. Las variables dummy (también conocidas como *ficticias* o *binarias*) tienen por finalidad transformar datos categóricos a numéricos utilizando una codificación binaria.

Por ejemplo, si dentro de la base de datos existe una columna llamada *países* ésta columna tendrá por entradas los nombres de los países, dígase México, EUA y Canadá. La transformación a variables dummy empieza por tomar cada entrada diferente de la columna categórica para convertirla en una columna más, solo con entradas binarias. Después, dentro de cada fila, se denota un 1 o un 0 si es que dicha fila de datos pertenecen a alguna de las nuevas columnas creadas (si, perteneciera a México, tendría un 1 en la fila correspondiente a la columna que representa a México y 0 en las dos otras columnas creadas).

Esta técnica resulta muy útil a la hora de querer realizar modelos de regresión con la existencia de variables categóricas en el modelo [37].

Coeficiente de Determinación (R^2)

Para el modelo de regresión lineal múltiple se tienen las siguientes definiciones de estadísticos:

- **Varianza total:** $\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$.
- **Varianza explicada:** $\frac{1}{n-1}\sum_{i=1}^{n}(\hat{y}_i - \bar{y}^2) = \frac{1}{n-1}(\hat{\boldsymbol{\beta}}'\mathbb{X}'\mathbf{Y} - n\bar{y}^2)$.
- **Varianza inexplicada:** $\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \hat{y}_i^2) = \frac{1}{n-1}(\sum_{i=1}^{n} y_i^2 - \hat{\boldsymbol{\beta}}'\mathbb{X}'\mathbf{Y})$.

El coeficiente de determinación es la proporción de la varianza total de la variable explicada por la regresión. Es también denominado R cuadrado y sirve para reflejar la bondad del ajuste de un modelo a la variable que se pretende explicar. Para calcular el *coeficiente de determinación* (R^2) se calcula el cociente de la varianza explicada con la varianza total, de modo que:

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = \frac{\hat{\boldsymbol{\beta}}'\mathbb{X}'\mathbf{Y} - n\bar{y}^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.14}$$

Este estadístico permite conocer con exactitud el grado de dispersión de los valores de una variable en relación con el resto [5] [6] [28].

Coeficiente de Determinación Ajustado

Aún cuando las variables independientes no estuvieran relacionadas, es posible que R^2 sea mayor a cero. Para prevenir una sobre valoración de la importancia de las variables se suele calcular el coeficiente de correlación ajustado (R^2-ajustado).

Las m variables aleatorias independientes explican lo suficiente la variación total de los valores observados para que, en promedio, $R^2 = \frac{m}{n-1}$ (esto sólo sí se cumple que las variables m sean, en efecto, aleatorias e independientes). El primer ajuste para el coeficiente de determinación es realizar la siguiente diferencia $R^2_{ajustado} = R^2 - \frac{m}{n-1}$.

Así, el R^2 ajustado (o coeficiente de determinación ajustado) se utiliza en la regresión múltiple para ver el grado de intensidad o efectividad que tienen las variables independientes en explicar la variable dependiente.

Si los valores de las variables independientes no son aleatorios por completo, el valor de $R^2_{ajustado}$ es igual, en promedio, a cero. Ya que, si $R^2 = 1$, entonces, $R^2_{ajustado} = 1 - \frac{m}{n-1} \neq 1$.

Para definir un $R^2_{ajustado}$, que sea igual a uno cuando R^2 es igual a uno, se necesita que $R^2 - \frac{m}{n-1}$ multiplique a $\frac{n-1}{n-m-1}$. Dando entonces el coeficiente de correlación ajustado:

$$R^2_{ajustado} = R^2_{adj} = \left(R^2 - \frac{m}{n-1}\right)\left(\frac{n-1}{n-m-1}\right). \tag{3.15}$$

Aunque estos estadísticos bastan para reconocer un buen ajuste del modelo, puede que este mismo no sea el mejor de todos. Es decir, el que se encuentre más cercano a la realidad. Para encontrar el modelo que mejor se ajuste a la realidad se plantean hipótesis respecto al problema.

Definición 10 (Pruebas de Hipótesis) *Sean Θ un espacio paramétrico, $\Theta_0 \subseteq \Theta$ y Θ_0^c su complemento tal que $\Theta_0 \cup \Theta_0^c = \Theta$. Se le denomina como **hipótesis** (H) a una conjetura sobre un parámetro θ de la población [33].*

Un juego de hipótesis se suele formular de la siguiente forma:

$$H_0 : \theta \in \Theta_0 \qquad v.s. \qquad H_a : \theta \in \Theta_0^c.$$

*donde a H_0 se le conoce como **hipótesis nula** y a H_a como **hipótesis alternativa** o complementaria.*

El problema de prueba de hipótesis consiste en decidir cual hipótesis dentro del juego es verdadera.

Al subconjunto del espacio muestral de la estadística de prueba para el cual la hipótesis nula se rechaza se le conoce como **región de rechazo**, mientras que a su complemento se le conoce como **región de no rechazo**.

Mejores Variables Regresoras

Dentro del análisis en la regresión lineal múltiple se pueden encontrar un número de variables independientes muy grande y no siempre todas las variables tiene el mismo *peso*. Es por ello que existen metodologías las cuales ayudan a encontrar el mejor modelo con las variables más significativas usando estadísticos que indiquen la significancia de las variables independientes

dentro del modelo. Estas metodologías se les suele llamar como *criterios de información estadística/estocástica* (CIE) [31].

Los *CIE* tienen la finalidad de identificar el mejor modelo de regresión de entre varios modelos candidatos. Los dos más utilizados son el criterio de información de *Akaike* (AIC) y el criterio de información Bayesiano (BIC).

El **AIC** proporciona una estimación de la distancia que existe entre el modelo y el mecanismo que genera los datos observados. La idea central es la penalización por un sobre ajuste del modelo, el cual está dado por:

$$\mathrm{AIC}(k) = 2k - 2\ln(\mathbf{l}(\hat{\Theta}(k))). \tag{3.16}$$

donde:

- $\mathbf{l}(\hat{\Theta}(k))$ es la función de verosimilitud.
- $\hat{\Theta}(k)$ es la estimación obtenida por máxima verosimilitud del parámetro θ.
- k es el número de parámetros libres estimados dentro del modelo.

Mientras que el criterio de información de Akaike no depende del tamaño de la muestra, el **criterio de información Bayesiano** (BIC), también conocido como el criterio de Schwarz, toma un enfoque bayesiano dónde se penaliza el número de parámetros con $\ln(n)$. De modo que el BIC está dado por:

$$\mathrm{BIC}(k) = k\ln(n) - 2ln(\mathbf{l}(\hat{\Theta}(k))). \tag{3.17}$$

3.3. Regresión Logística

El modelo de regresión logística es parte del aprendizaje supervisado y es usado en el aprendizaje automático como un método de clasificación. Esta regresión contempla las variables dependientes $Y_1, \ldots, Y_n$; $n \in \mathbb{N}$ distribuidas cada una forma $Bernoulli(\lambda_i)$ de forma independiente en dónde $\mathbb{E}[Y_i] = \mathbb{P}(Y_i = 1) = \lambda_i$ para todo i [28]. En este modelo λ_i se relaciona con las variables independientes del modelo de regresión x_i de la siguiente forma:

$$\log\left(\frac{\lambda_i}{1-\lambda_i}\right) = \beta_0 + \beta_1 x_i. \tag{3.18}$$

El logaritmo de probabilidades (o *logit*) es una función lineal de la variable (predictora) x. Tomando la función de densidad de una variable aleatoria $\mathbf{B}ernoulli(\lambda)$ en su forma exponencial.

$$f(y) = (1-\lambda_i)\exp\left\{y\log\left(\frac{\lambda_i}{1-\lambda_i}\right)\right\}.$$

Dentro de la función de densidad se encuentra la función $g(\lambda_i) = \log\left(\frac{\lambda_i}{1-\lambda_i}\right)$ la cual se utiliza como *función de enlace* para reescribir a la Ecuación (3.18) de la siguiente forma:

$$\lambda_i = \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}.$$

y de forma más general[1]

$$\lambda(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}. \tag{3.19}$$

Respecto a la función $\lambda(x)$ se notan tres propiedades en concreto:

[1] Es posible que $\lambda(x) = 0$ o $\lambda(x) = 1$; en dado caso se dice que el modelo de regresión logística no es el adecuado para el problema.

1. Si $\beta_1 < 0$, $\lambda(x)$ es monótona decreciente.
2. Si $\beta_1 > 0$, $\lambda(x)$ es monótona creciente.
3. Si $\beta_1 = 0$, $\lambda(x) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$ para todo x.

Dentro de este modelo de regresión la relación que existe entre λ y x es

$$\lambda\left(-\frac{\beta_0}{\beta_1}\right) = \frac{1}{2}.$$

Lo cual muestra un tipo de simetría dentro de la función. Además de que, dado un $c \in \mathbb{R}$ cualquiera:

$$\lambda\left(-\frac{\beta_0}{\beta_1} + c\right) = 1 - \lambda\left(-\frac{\beta_0}{\beta_1} - c\right). \tag{3.20}$$

De modo que, evaluando en x y a $x+1$ en la Ecuación (3.18), tomando a la Ecuación (3.19) para realizar la siguiente diferencia se tiene:

$$\log\left(\frac{\lambda(x+1)}{1-\lambda(x+1)}\right) - \log\left(\frac{\lambda(x)}{1-\lambda(x)}\right) = \beta_0 + \beta_1(x+1) - (\beta_0 + \beta_1 x) = \beta_0 + \beta_1 x + \beta_1 - \beta_0 - \beta_1 x = \beta_1.$$

De modo que:

$$\log\left(\frac{\lambda(x+1)}{1-\lambda(x+1)}\right) - \log\left(\frac{\lambda(x)}{1-\lambda(x)}\right) = \beta_1. \tag{3.21}$$

para cualquier valor de x.

De este modo se dice que β_1 es el *cambio en probabilidades de éxito logarítmicos* de forma correspondiente a x en aumentos unitarios. Calculando la exponencial en ambos lados de la ecuación se obtiene que:

$$e^{\beta_1} = \frac{\lambda(x+1)/\left[1-\lambda(x+1)\right]}{\lambda(x)/\left[1-\lambda(x)\right]}. \tag{3.22}$$

El lado derecho de la Ecuación (3.22) se conoce como la *razón de probabilidades* la cual hace comparación con las probabilidades de éxito en $x+1$ con respecto a las probabilidades de éxito en x. Por último se tiene que:

$$\frac{\lambda(x+1)}{1-\lambda(x+1)} = e^{\beta_1}\frac{\lambda(x)}{1-\lambda(x)}. \tag{3.23}$$

El cual dice que e^{β_1} es el cambio multiplicativo en las probabilidades de éxito correspondiente a un aumento de x en una unidad.

3.3.1. Estimación

Dentro de las maneras que existen para estimar un modelo de regresión logística, el más ocupado es calculando los estimadores de máxima verosimilitud ya que, en general, contamos con una sucesión de variables aleatorias independientes cada una con la misma distribución (iid) $Y_i \sim Bernoulli(\lambda_i)$ cada una con su respectiva función de distribución $F_i = F(\beta_0 + \beta_1 x_i)$ [5] [28]. La función de verosimilitud está dada por:

$$\mathbf{L}\left(\beta_0, \beta_1; \mathbf{y}\right) = \prod_{i=1}^{n} F_i^{y_i}(1-F_i)^{1-y_i}. \tag{3.24}$$

Del mismo modo, el logaritmo de la función de verosimilitud $l(\beta_0, \beta_1)$ está dado por:

$$l(\beta_0, \beta_1) = \sum_{i=1}^{n} \left\{ \log(1 - F_i) + y_i \log\left(\frac{F_i}{1 - F_i}\right) \right\}. \tag{3.25}$$

Derivando a la función $l(\beta_0, \beta_1)$ respecto a cada uno de los argumentos, se tiene que:

$$\frac{\partial l(\beta_0, \beta_1)}{\partial \beta_0} = \sum_{i=1}^{n} (y_i - F_i) \frac{f_i}{F_i(1 - F_i)} \quad \text{y} \quad \frac{\partial l(\beta_0, \beta_1)}{\partial \beta_1} = \sum_{i=1}^{n} (y_i - F_i) \frac{f_i}{F_i(1 - F_i)} x_i.$$

Al ser la regresión logística un modelo no lineal, los parámetros beta's tienen que ser calculados de manera numérica. Pero, en caso de tener una regresión logística con $F(x) = \frac{e^x}{1+e^x}$, $\frac{f_i}{F_i(1-F_i)} = 1$ las ecuaciones de arriba se simplifican y hacen más fácil el cálculo para los estimadores de máxima verosimilitud de manera usual.

Dado que se suelen aplicar varios modelos para encontrar el más parsimonioso (o el de mejor ajuste con el menor número posible de variables), se hace uso de los conceptos de *sensibilidad* y *especificidad* dentro de la definición de *Matriz de Confusión* además de definir las *Curvas de ROC*. A este tipo de métodos para medir el ajuste del modelo se conocen como *Métodos de Evaluación de Rendimiento* o *Evaluación de Algoritmos de Aprendizaje* [12].

3.4. Métodos de Evaluación de Rendimiento

Existe una cantidad considerable de métodos para evaluar el desempeño de el o los modelos implementados. El método fundamental es la *matriz de confusión*, la cual está conformada por los valores totales que se obtuvieron de manera satisfactoria en comparación con los que erraron. Es decir, la matriz de confusión denota la cardinalidad de aciertos que obtuvo el clasificador en comparación con los fallos.

3.4.1. Matriz de Confusión

Definición 11 (Matriz de Confusión) *Sean $\mathbb{M}_{n\times n}$ el conjunto de matrices de orden $n \times n$, $\mathbf{C} \in \mathbb{M}_{n\times n}$, f un clasificador fijo[2], T un conjunto de datos y $l \in \mathbb{N}$ el número de clases; se denota a la matriz de confusión respecto al clasificador f como $\mathbf{C}(f)$. Cada entrada $c_{ij}(f)$ representa el total de elementos que pertenecen a la clase i y que f los clasifica en la clase j. Es decir, el elemento $c_{15}(f)$ representa el total de elementos que pertenecen a la clase 1 y que f asigna a la clase 5 [12]. De modo que, la matriz de confusión se expresa como:*

$$\mathbf{C}(f) = \{c_{ij}(f)\} = \left\{ \sum_{x \in T} [y = i] \wedge [f(x) = j] \right\}. \tag{3.26}$$

donde la variable y representa la clase inicial y x a los elementos dentro de T.

De la definición se sigue con los enunciados:

- $\sum_{j=1}^{l} c_{ij}(f) = c_{i.}(f)$ es el total de elementos dentro de T que pertenecen a la clase i.
- $\sum_{i=1}^{l} c_{ij}(f) = c_{.j}(f)$ es el total de elementos dentro de T que fueron asignados a la clase j por medio del clasificador f.

[2]Clasificador Fijo: Cualquier algoritmo que se use para clasificar (k-vecinos más cercanos, árboles de decisión, etc.) que se utilizó para el modelo paramétrico.

- Los valores de la diagonal principal representan a los elementos clasificados correctamente a la clase i. De modo que:
  $\sum_{i=1}^{l} c_{ii}(f)$ es el total de elementos clasificados de manera correcta mediante el clasificador f.
- Los valores fuera de la diagonal principal representan las clasificaciones erróneas de f. Por lo tanto:
  $\sum_{i,j=1:j\neq i} c_{ij}(f)$ es el total de elementos clasificados de manera errónea mediante f.

Detección de Aciertos y Fallos

Para medir el rendimiento de implementación del modelo es necesario medir la *tasa de aciertos* y la *tasa de fallos*. La primer tasa indica la proporción de individuos de una clase en particular asignados correctamente mediante el clasificador f a la misma clase inicial. Es decir, para los individuos en la clase i, el clasificador f los asigna correctamente en la clase inicial i. Mientras que la *tasa de fallos* indica la proporción de individuos de una clase en particular asignados de manera errónea mediante el clasificador f.

Para conocer de manera veraz la proporción correcta de aciertos y fallos es necesario clasificar los resultados en *verdaderos positivos* y en *falsos positivos* mediante la siguiente definición.

Definición 12 (Tasa de Verdaderos Positivos y Tasa de Verdaderos Negativos) *Sea $\boldsymbol{C}(f)$ la matriz de confusión con l clases y f un clasificador fijo, la tasa de verdaderos positivos está dada por:*

$$TVP_i(f) = \frac{c_{ii}(f)}{\sum_{j=1}^{l} c_{ij}(f)} = \frac{c_{ii}(f)}{c_{i.}(f)}. \tag{3.27}$$

Y la tasa de verdaderos negativos está dada por:

$$TVN_i(f) = \frac{\sum_{j:j\neq i} c_{ji}(f)}{\sum_{j,k:j\neq i} c_{jk}(f)}. \tag{3.28}$$

Por lo tanto, la $TVP_i(f)$ mide la proporción de elementos que pertenecen a la clase i y que se clasifican de manera correcta a la clase i. Por el contrario, la $TVN_i(f)$ mide la proporción de elementos que no pertenecen a la clase i y que se clasifican de manera errónea a esta clase.

Caso Binario

El caso más común al utilizar la matriz de confusión es cuando $l = 2$; estas clases se conocen como *positivo* y *negativo* respectivamente. Las filas de la matriz representan la clase inicial de los elementos y las columnas la clase asignada por el clasificador f. La matriz de confusión binaria contiene cuatro valores de importancia: Total de verdaderos positivos, total de falsos positivos, total de verdaderos negativos y el total de falsos negativos [10].

Así el **total de negativos (N)** y el **total de positivos (P)** se obtiene mediante las sumas:

$$\mathbf{N} = VerdaderosNegativos(VN) + FalsosPositivos(FP)$$
$$\mathbf{P} = FalsosNegativos(FN) + VerdaderosPositivos(VP).$$

Las cuales se utilizan para calcular las cuatro tasas siguientes:

- Tasa de verdaderos positivos:

$$\text{TVP}(f) = \frac{VP}{VP + FN} = \frac{c_{22}(f)}{c_{21}(f) + c_{22}(f)}.$$

- Tasas de falsos positivos:

$$\text{TFP}(f) = \frac{FP}{FP + VN} = \frac{c_{12}(f)}{c_{11}(f) + c_{12}(f)}.$$

- Tasa de verdaderos negativos:

$$\text{TVN}(f) = \frac{VN}{VN + FP} = \frac{c_{11}(f)}{c_{11}(f) + c_{12}(f)}.$$

- Tasa de falsos negativos:

$$\text{TFN}(f) = \frac{FN}{FN + VP} = \frac{c_{21}(f)}{c_{21}(f) + c_{22}(f)}.$$

3.4.2. Curvas de ROC

El análisis de *Características Operativas del Receptor* (ROC por sus siglas en inglés) es una técnica gráfica para los modelos de clasificación. Esta curva está formada por puntos de corte los cuales se utilizan para conocer la precisión de los modelos generados a partir del puntaje que se obtuvo de cada uno.

Esta curva particiona el plano en dos conjuntos. Dicho plano se conoce como *Espacio de ROC* [10].

Espacio de ROC

Dado un conjunto de datos y cualquier modelo de regresión, ϵ_i denota el error de cada instancia i. Dichos errores toman valores positivos como negativos y, para la búsqueda del mejor modelo, se define la *Sobre Estimación Total* y la *Sub Estimación Total.*

Definición 13 (Sobre Estimación y Sub Estimación Total) *Sea ϵ_i el error de la instancia i, se define la Sobre Estimación Total y la Sub Estimación Total como sigue:*

1. *La Sobre Estimación Total es la suma de todos los errores mayores a cero, es decir:*

$$OVER = \sum_i \{\epsilon_i \mid \epsilon_i > 0\}.$$

2. *La Sub Estimación Total es, de manera análoga, la suma de todos los errores menores a cero:*

$$UNDER = \sum_i \{\epsilon_i \mid \epsilon_i < 0\}.$$

Definición 14 (Espacio de ROC) *El espacio de características operativas del receptor de regresión (RROC, por sus siglas en inglés) se define como un subespacio de $\mathbb{R}^2$ resultado del producto cartesiano de $OVER$ y $UNDER$. Es decir, $OVER \times UNDER = RROC$, con $OVER \in \mathbb{R}^+$ y $UNDER \in \mathbb{R}^-$. Esto hace que, de manera gráfica, se coloque el origen en la esquina superior izquierda; adquiriendo entonces el nombre de el cielo de RROC [10].*

Notar que, dependiendo del estimador que se use para medir el $OVER$ y $UNDER$, sus valores no necesariamente son con signos complementarios; es decir, si en vez de ϵ_i se utiliza alguna otra estimación se debe de construir el espacio a partir de los conjuntos donde se arrojan los valores para $OVER$ y $UNDER$. Más adelante se ilustra mediante la construcción.

RROC es particionado mediante la curva de ROC, cuya construcción se hace con los valores de $OVER$ y $UNDER$ para diferentes modelos aplicados a la población. La teoría que abarca el estudio de las curvas de ROC es bastante extensa, por lo que se explica la construcción dentro de un ejemplo con Python.

3.5. Ejemplo con Python

3.5.1. Regresión Lineal Simple

A continuación se presenta una aplicación del lenguaje Python utilizado como software estadístico para analizar las correlaciones del *Peso* contra las demás características de los individuos. Esto con el propósito de encontrar las variables independientes que tengan una mayor correlación con dicha medida.

Se cuenta con una muestra de 252 individuos masculinos donde se especifican, para cada uno, los valores de:

- Densidad.
- Porcentaje de grasa corporal.
- Edad.
- Peso.
- Altura.
- Circunferencia del cuello.
- Circunferencia del pecho.
- Circunferencia del abdomen.
- Circunferencia del muslo.
- Circunferencia de la rodilla.
- Circunferencia del tobillo.
- Circunferencia del bíceps.
- Circunferencia del antebrazo.
- Circunferencia de la muñeca.

Con el objetivo de determinar el mejor juego de variables que explique al peso, mediante un modelo de regresión lineal.

Respecto a la regresión lineal simple, un primer modelo que se plantea es explicar el peso mediante la circunferencia del abdomen; esto debido a la relación que existe entre el peso y la expansión del mismo abdomen. En la regresión múltiple el peso se explica utilizando todas las características que otorga la base.

Esta base de datos fue obtenida de *kaggle.com* [14] y el código con el catálogo de funciones se encuentra en el Capítulo 8 de Anexos.

Bibliotecas

[2]:
```
import pandas as pd # Paquetería para cargar archivos
import Catalogo_de_Funciones as fun # Paquetería del catálogo de funciones
```

Exploración

Se comienza por extraer pocos valores de la base de datos para conocer como está constituida y generar estadísticos globales para un primer acercamiento de los valores que formarán parte del modelo como se muestra a continuación:

[2]:
```
path = 'GrasaCorporal.xlsx'
fun.head_describe_xlsx(path)
```

[2]:

Índice	Densidad	GrasaCorporal	Edad	Peso	Altura
0	1.07	0.12	23	69.9	1.72
1	1.08	0.06	22	78.5	1.84
2	1.04	0.25	22	69.8	1.68
3	1.07	0.1	26	83.8	1.84
4	1.03	0.28	24	83.5	1.81
Índice	CirCuello	CirPecho	CirAbdómen	CirCadera	CirMuslo
0	36.2	36.2	85.2	94.5	59
1	38.5	38.5	83	98.7	58.7
2	34	34	87.9	99.2	59.6
3	37.4	37.4	86.4	101.2	60.1
4	34.4	34.4	100	101.9	63.2
Índice	CirRodilla	CirAntebrazo	CirMuñeca	CirTobillo	CirBíceps
0	37.3	27.4	17.1	21.9	32
1	37.3	28.9	18.2	23.4	30.5
2	38.9	25.2	16.6	24	28.8
3	37.3	29.4	18.2	22.8	32.4
4	42.2	27.7	17.7	24	32.2

Tabla 3.1: Extracción de tamaño cinco de la base de datos.

Este código genera dos tablas, la Tabla 3.1 muestra los valores de los primeros cinco individuos (identificados con 0,1,2,3 y 4) con sus respectivas características y los valores asignados a cada individuo. La finalidad es mostrar, a grandes rasgos, la composición de la base de datos que se tiene. A este paso, generalmente, se le conoce como *comprensión o entendimiento de los datos*.

En una primera observación, se nota que los valores de la circunferencia de rodilla, de antebrazo y de muñeca no distan mucho entre sí; al contario del peso o la altura. Además es de notar que los valores de la circunferencia del cuello y el pecho parecen ser los mismos.

Pero, como esta extracción es muy pequeña, no se puede concluir nada acerca de los valores y su relación entre ellos. Para realizar las primeras suposiciones es de interés conocer estadísticos generales de la base de datos.

Estadísticos como la desviación estándar, los valores cuartiles e incluso la media resultan de mucha ayuda para conocer más acerca del comportamiento de la base.

Por ende, se calcula la segunda tabla para conocer los estadísticos de conteo, media, desviación estándar, valor mínimo, Q_1, Q_2, Q_3 y el valor máximo de cada característica.

[2]:

Estadístico	Densidad	GrasaCorporal	Edad	Peso	Altura
count	252	252	252	252	252
mean	1.06	0.19	44.88	81.16	1.78
std	0.02	0.08	12.60	13.33	0.09
min	0.995	0	22	53.75	0.75
25\%	1.04	0.12	35.75	72.12	1.73
50\%	1.05	0.19	43	80.06	1.78
75\%	1.07	0.25	54	89.36	1.84
max	1.11	0.48	81	164.72	1.97
Estadístico	CirCuello	CirPecho	CirAbdomen	CirCadera	CirMuslo
count	252	252	252	252	252
mean	37.99	37.99	92.56	99.90	59.41
std	2.43	2.43	10.78	7.16	5.25
min	31.1	31.1	69.4	85	47.2
25\%	36.4	36.4	84.58	95.5	56
50\%	38	38	90.95	99.3	59
75\%	39.43	39.43	99.33	103.53	62.35
max	51.2	51.2	148.1	147.7	87.3
Estadístico	CirRodilla	CirTobillo	CirBíceps	CirAntebrazo	CirMuñeca
count	252	252	252	252	252
mean	38.59	23.10	32.27	28.66	18.23
std	2.41	1.69	3.02	2.02	0.93
min	33	19.1	24.8	21	15.8
25\%	36.98	22	30.2	27.3	17.6
50\%	38.5	22.8	32.05	28.7	18.3
75\%	39.93	24	34.33	30	18.8
max	49.1	33.9	45	34.9	21.4

Tabla 3.2: Tabla con estadísticos generales de la base de datos.

La Tabla 3.2 muestra los estadísticos de conteo, media, desviación estándar, valor mínimo, cuartiles y el valor máximo para cada característica de la base de datos. En esta primer exploración con los datos se nota que los valores para la circunferencia del cuello y del pecho son prácticamente idénticos, haciéndose notar una dependencia lineal la cual afectaría al modelo de forma contraproducente. De modo que se calcula una *matriz de correlación* para conocer la influencia de las variables entre sí. Además de que se espera que la correlación entre la circunferencia del cuello y del pecho sea igual a uno.

Matriz de Correlación

La matriz de correlación funciona perfecto para conocer que tan cerca están entre sí de tener una relación lineal. Es decir, mientras más cercano sea el valor de la correlación a uno o a menos uno más fuerte es la relación lineal entre dichas variables. De hecho, mientras el valor se acerque más a -1 se dice que dichas variables tienen una relación proporcional inversa; es decir que mientras una aumenta la otra disminuye. De este modo se buscan los valores que mejor expliquen el experimento.

```
[6]: data = pd.read_excel('GrasaCorporal.xlsx')
     fun.MatrizCorrelación(data)
```

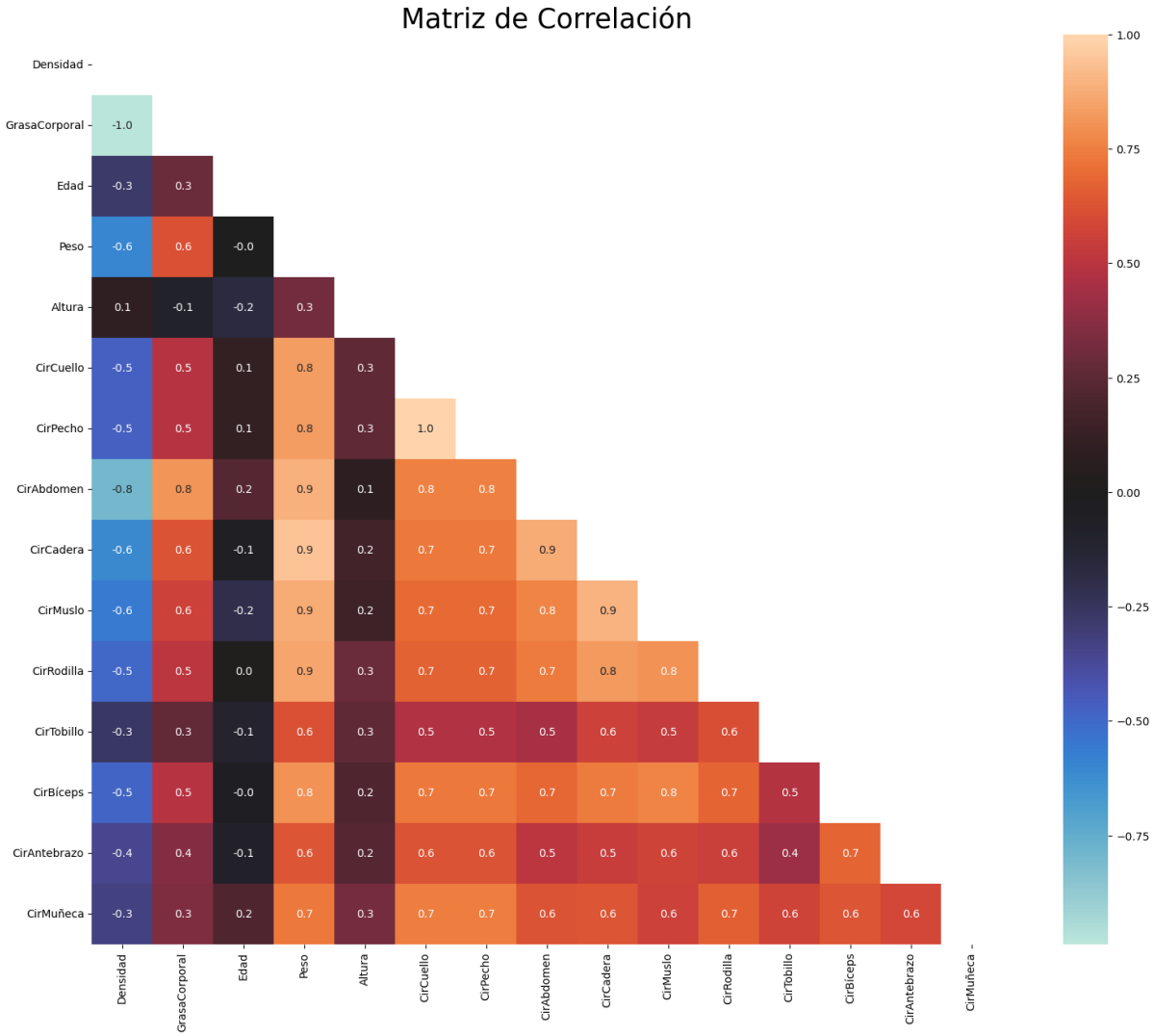

Figura 3.1: Matriz de Correlación

Como se muestra en la Figura 3.1, las características de *CirCuello* y *CirPecho* tienen una alta correlación (igual a 1), por consiguiente sólo se selecciona una de las dos variables para evitar un sobre ajuste y multicolinealidad. Se observa que la variable de *Densidad* tiene un comportamiento proporcional inverso sobre todas las demás características (sin contar la altura). Pero, por el momento, es de interés conocer el peso del individuo en términos de una única variable para el modelo de regresión lineal simple y la *densidad* no califica para este modelo en particular. De modo que los tres mejores candidatos para el modelo son:

- Circunferencia de Cadera (con una puntuación de 0.94).
- Circunferencia del Abdomen (con una puntuación de 0.89).
- Circunferencia del Muslo (con una puntuación de 0.87).

Cada una de estas relaciones se interpretan de manera gráfica de la siguiente manera:

[7]:
```
# Extracción de las características a utilizar
Top = data[['CirCadera','CirAbdomen','CirMuslo', 'Peso']]

# Selección de nombres de columnas
X_col = Top.columns[:-1]
y_col = Top.columns[-1:]

# Visualización de diagramas de dispersión
fun.MatrizDiagramasDispersion(2,2,X_col,y_col,data)
```

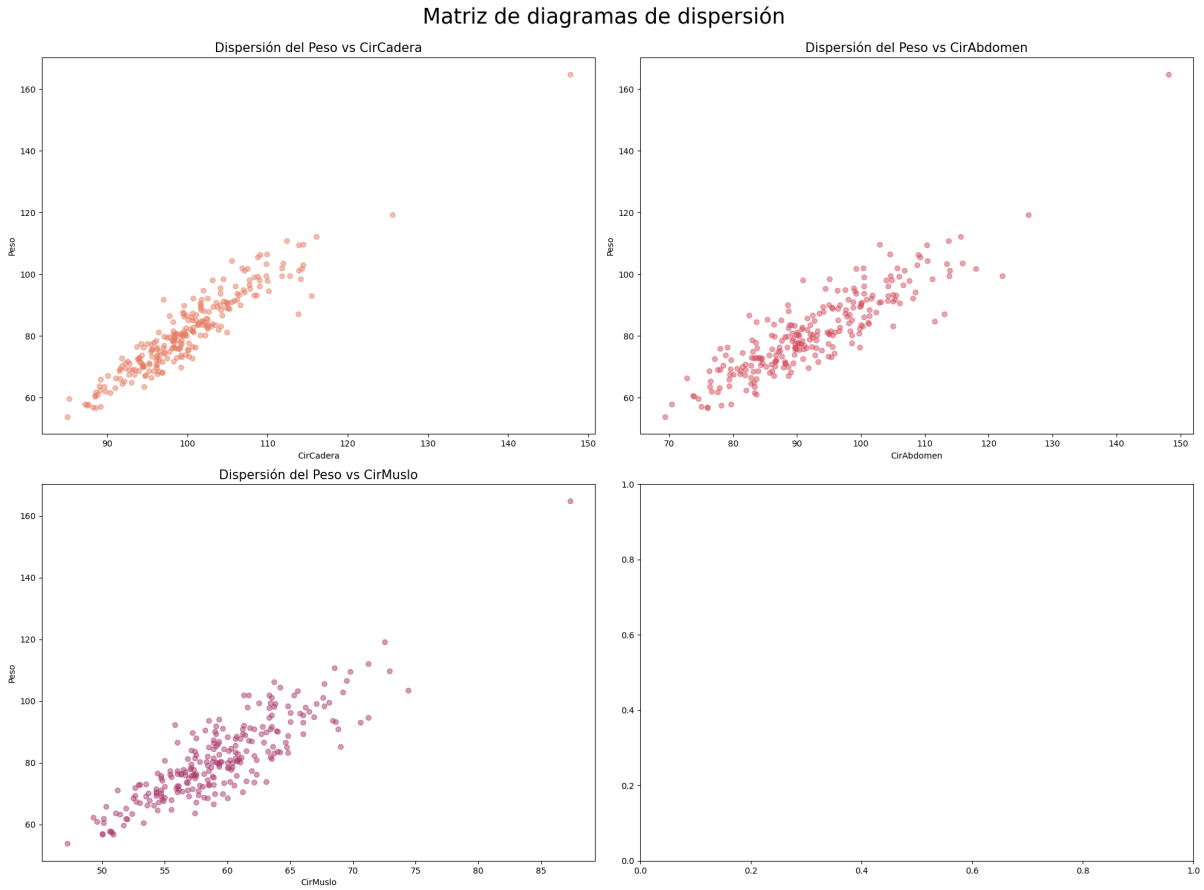

Figura 3.2: Gráficos de Dispersión de cada Regresión Simple.

La Figura 3.2 muestra (de arriba hacia abajo) tres diagramas de dispersión en donde se relaciona la variable de *peso* (en el eje vertical) con cada una correspondiente a su título (en el eje horizontal).

Las tres características denotan un comportamiento lineal con respecto al peso. Para conocer cuál es la que mejor explica el peso, se continua con la modelación de las regresiones.

Preparación de los Datos

Como se van a comparar tres regresiones lineales simples, se comienza por escoger los datos para que cada implementación tenga sólo las que van a participar en la regresión. Es necesario

identificar cuál va a ser la variable independiente X y la variable dependiente y para cada modelo. De modo que se denotan de la siguiente forma:

- `X`: variable independiente del modelo.
- `y`: variable objetivo (o dependiente) del modelo circunferencia de cadera vs peso. Asignada al peso.
- `X_col`: variable que denota el nombre de las columnas en la tabla de `Top` a utilizar en cada regresión (se utiliza para graficar).

Es de notar que, dentro de la asignación de variables para las regresiones, se realiza una partición de la muestra; denotada por "prueba" y "entrenamiento". Dichas particiones son tomados de manera aleatoria y el tamaño de prueba corresponde a $\frac{1}{3}$ del total de la base de datos original. Una condición que se debe de cumplir es que el conjunto de prueba debe de ser menor (en cardinalidad) al de entrenamiento. Dicho tamaño de la partición se realiza dentro de las funciones declaradas en el catálogo y, por supuesto, en caso de que se requiera cambia utilizando el argumento que lo denote.

Con esto ya se puede dar pie a la implementación de los modelos de regresión. Para la implementación de un modelo de regresión lineal simple, se utiliza el proceso por máxima verosimilitud así como las paqueterías de Python para aplicar aprendizaje automático.

De este modo, la implementación utilizando máxima verosimilitud es como sigue:

Máxima Verosimilitud

Circunferencia de cadera vs. peso

[9]:
```
# Selección de características
X = Top[['CirCadera']].values
y = Top[['Peso']].values

# Selección de columna
X_col = Top.columns[:-1]

# Aplicación y visualización del modelo
fun.MaxVerRLS(X,y)
fun.MVTestDisperGraf(X_col,y_col,X,y)
fun.MVTrainDisperGraf(X_col,y_col,X,y)
```

Estadísticos de la regresión

```
Valor del intercepto: -88.47.
Valor del coeficiente: 1.70.
Primeros cinco valores de la regresión: [56.41, 82.57, 89.87, 78.32, 124.86].
Coeficiente de Determinación: 89.01%.
```

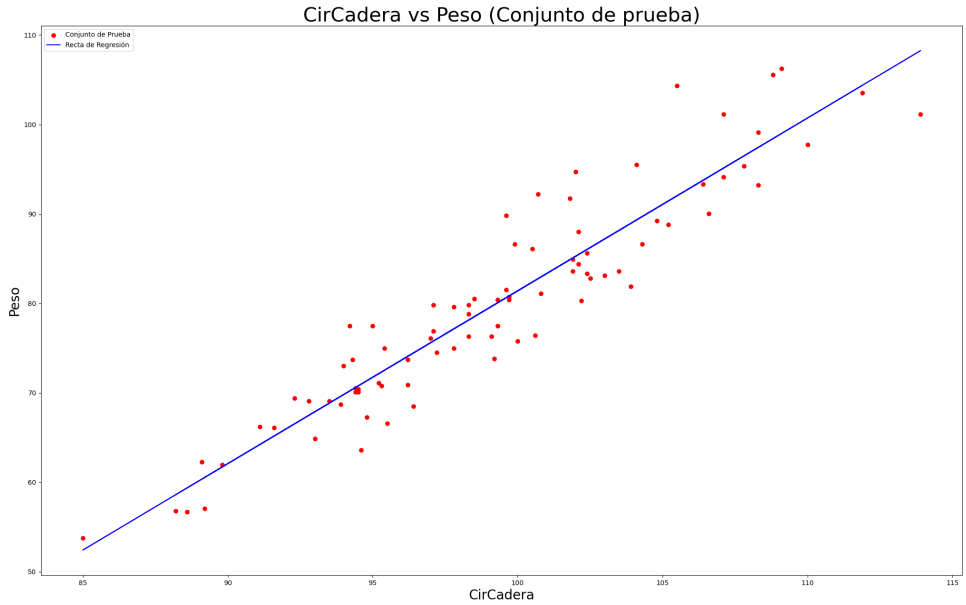

Figura 3.3: Gráfico de dispersión (rojo) y recta de regresión (azul) del conjunto de prueba.

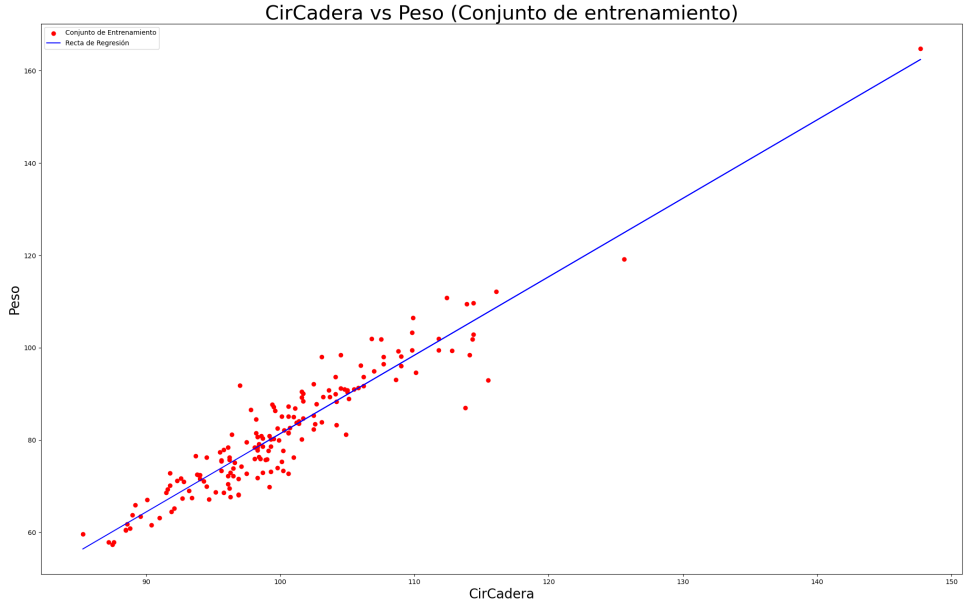

Figura 3.4: Gráfico de dispersión (rojo) y recta de regresión (azul) del conjunto de entrenamiento.

El primer modelo de regresión lineal simple paramétrico (Circunferencia de cadera vs Peso) tiene entonces la siguiente forma:

$$y = -88.47 + 1.7x.$$

En donde y representa el valor de la variable dependiente utilizando el valor de la variable independiente x. Es decir, conociendo el valor de x (circunferencia de cadera) se pronostica del peso y.

Por otro lado, el modelo de estimación tiene la forma:

$$\hat{y}_i = -88.47 + 1.7x_i.$$

$\hat{y}_i$ representa el valor de la predicción utilizando los valores de la variable dependiente x_i sobre toda la muestra.

Circunferencia de abdomen vs. peso

[11]:
```
# Selección de características
X_2 = Top[['CirAbdomen']].values
y = Top[['Peso']].values

# Selección de columna
X_col_2 = Top.columns[1:2]

# Aplicación y visualización del modelo
fun.MaxVerRLS(X_2,y)
fun.MVTestDisperGraf(X_col_2,y_col,X_2,y)
fun.MVTrainDisperGraf(X_col_2,y_col,X_2,y)
```

Estadísticos de la regresión

```
Valor del intercepto: -20.82.
Valor del coeficiente: 1.10.
Primeros cinco valores de la regresión: [61.33, 85.66, 89.41, 84.12, 118.15].
Coeficiente de Determinación: 77.70%.
```

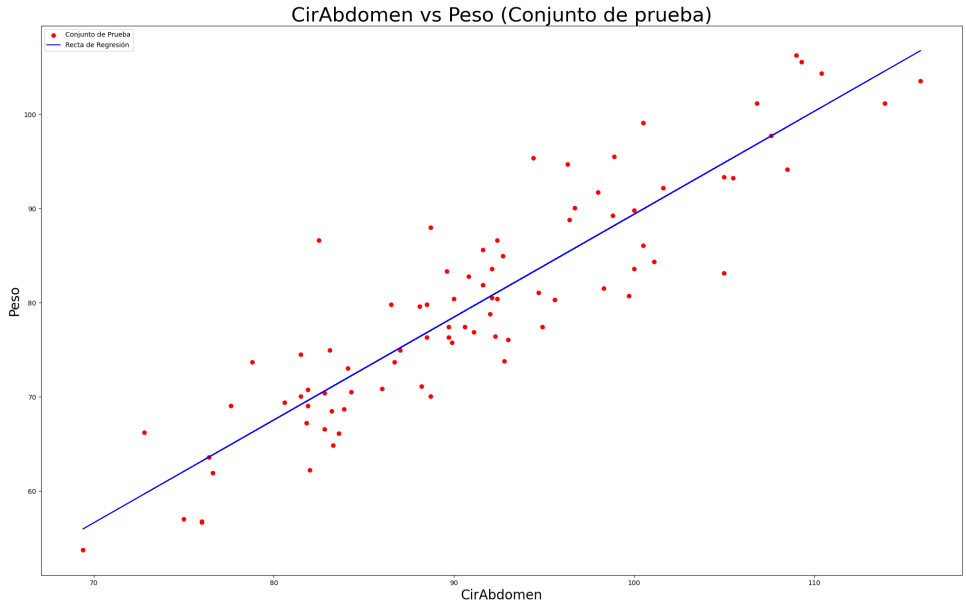

Figura 3.5: Gráfico de dispersión (rojo) y recta de regresión (azul) del conjunto de prueba.

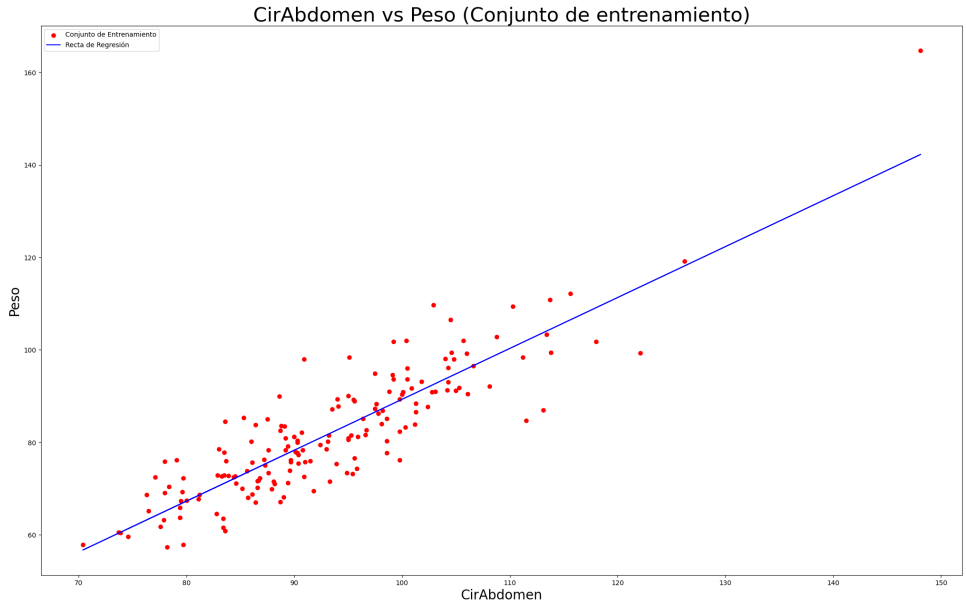

Figura 3.6: Gráfico de dispersión (rojo) y recta de regresión (azul) del conjunto de entrenamiento.

El segundo modelo de regresión lineal simple paramétrico (Circunferencia de abdomen vs Peso) tiene entonces la siguiente forma:

$$y = -20.82 + 1.1x.$$

En donde y representa el valor de la variable dependiente utilizando el valor de la variable independiente x. Es decir, conociendo el valor de x (circunferencia de abdomen) se pronostica del peso y.

Por otro lado, el modelo de estimación tiene la forma:

$$\hat{y} = -20.82 + 1.1x_i.$$

$\hat{y}_i$ representa el valor de la predicción utilizando los valores de la variable dependiente x_i sobre toda la muestra.

Circunferencia de muslo vs Peso

[13]:
```
# Selección de características
X_3 = Top[['CirMuslo']].values
y = Top[['Peso']].values

# Selección de columna
X_col_3 = Top.columns[2:3]

# Aplicación y visualización del modelo
fun.MaxVerRLS(X_3,y)
fun.MVTestDisperGraf(X_col_3,y_col,X_3,y)
fun.MVTrainDisperGraf(X_col_3,y_col,X_3,y)
```

Estadísticos de la regresión

```
Valor del intercepto: -52.91.
Valor del coeficiente: 2.27.
Primeros cinco valores de la regresión: [64.41, 81.65, 88.00, 77.34, 111.60].
Coeficiente de Determinación: 78.95%.
```

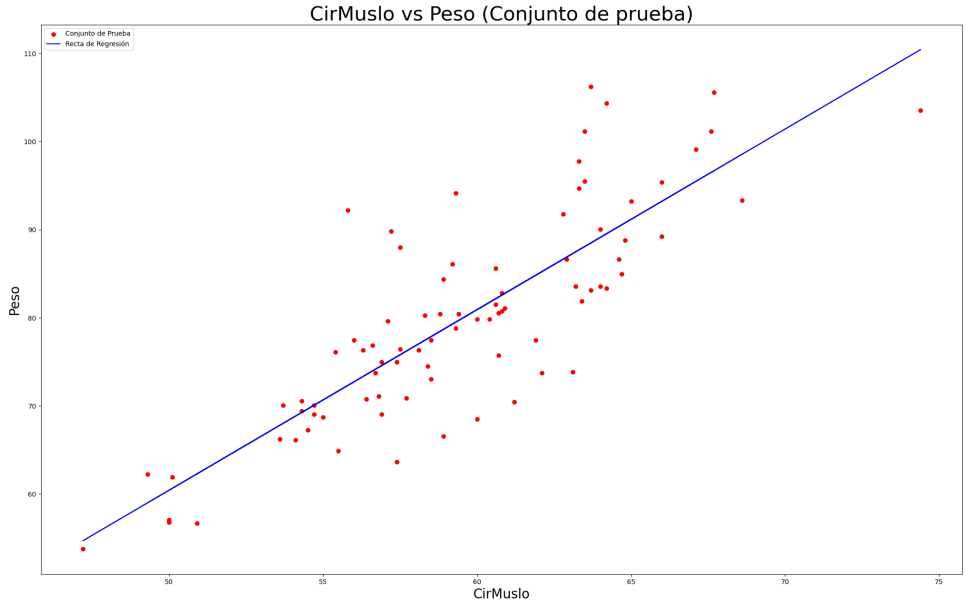

Figura 3.7: Gráfico de dispersión (rojo) y recta de regresión (azul) del conjunto de prueba.

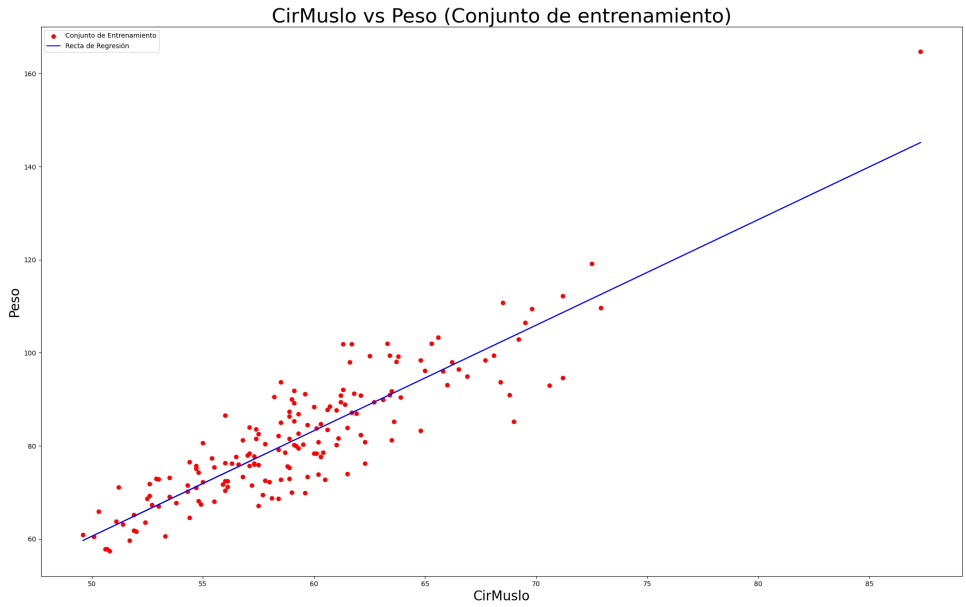

Figura 3.8: Gráfico de dispersión (rojo) y recta de regresión (azul) del conjunto de entrenamiento.

El tercer modelo de regresión lineal simple paramétrico (Circunferencia de muslo vs Peso) tiene entonces la siguiente forma:

$$y = -52.91 + 2.27x.$$

En donde y representa el valor de la variable dependiente utilizando el valor de la variable independiente x. Es decir, conociendo el valor de x (circunferencia de muslo) se pronostica del peso y.

Por otro lado, el modelo de estimación tiene la forma:

$$\hat{y} = -52.91 + 2.27x_i.$$

$\hat{y}_i$ representa el valor de la predicción utilizando los valores de la variable dependiente x_i sobre toda la muestra.

Aprendizaje Automático

Circunferencia de cadera vs. peso

[15]:
```
# Selección de características
X = Top[['CirCadera']].values
y = Top[['Peso']].values

# Selección de columna
X_col = Top.columns[:-1]

# Aplicación y visualización del modelo
fun.AARLS(X,y)
fun.AATrainDisperGraf(X_col, y_col, X, y)
fun.AATestDisperGraf(X_col,y_col,X,y)
```

Estadísticos de la regresión

```
Valor del intercepto: -88.47.
Valor del coeficiente: 1.70.
Primeros cinco valores de la regresión: [56.41, 82.57, 89.87, 78.32, 124.86].
Coeficiente de Determinación: 89.01%.
```

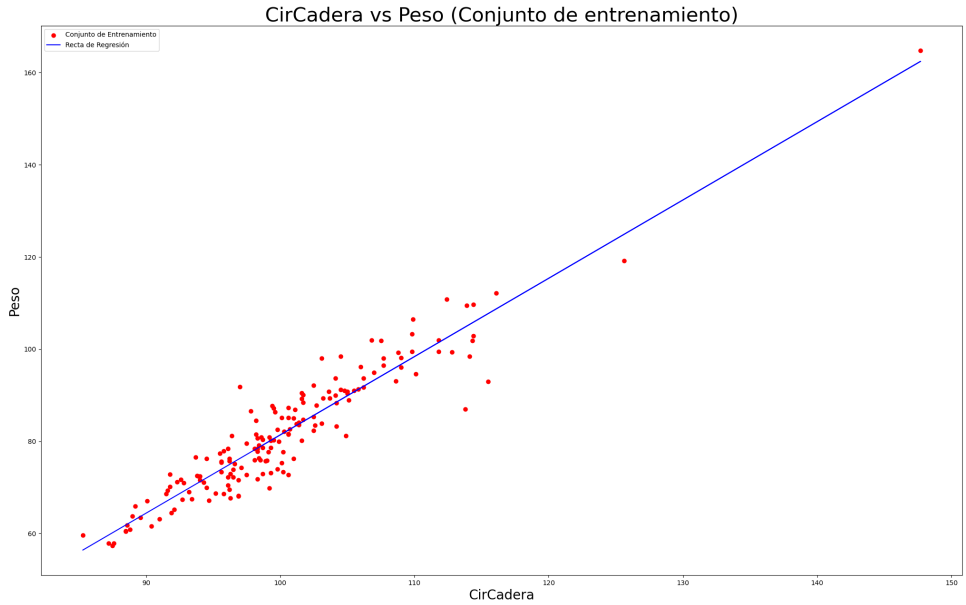

Figura 3.9: Gráfico de dispersión (rojo) y recta de regresión (azul) del conjunto de entrenamiento.

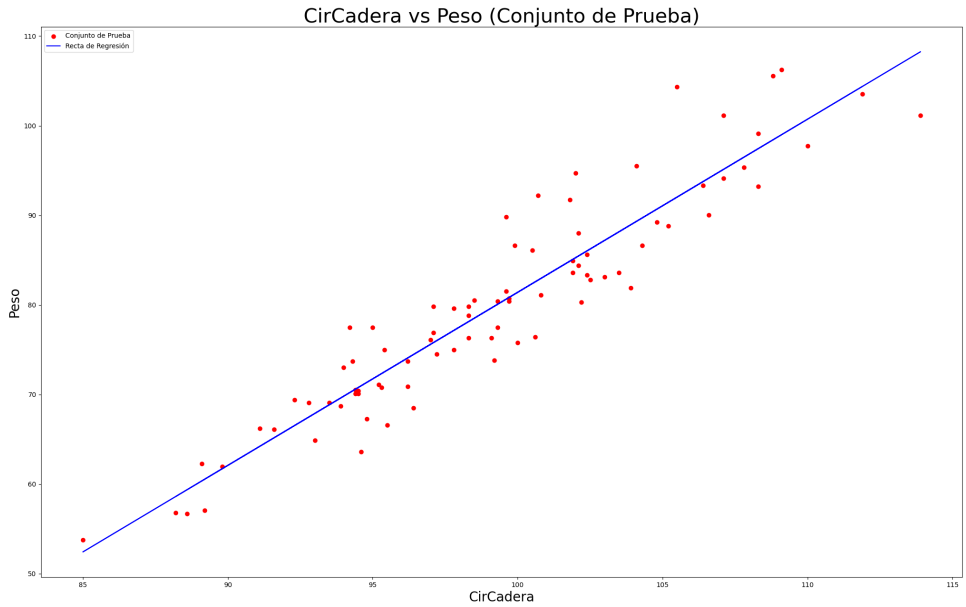

Figura 3.10: Dispersión de los valores de prueba (en rojo) contrastada con la recta de regresión generada (en azul).

El modelo de regresión lineal simple paramétrico (Circunferencia de cadera vs Peso), utilizando aprendizaje automático, tiene entonces la siguiente forma:

$$y = -88.47 + 1.7x.$$

Y, el modelo de estimación, tiene la siguiente forma:

$$\hat{y}_i = -88.47 + 1.7x_i.$$

Circunferencia de abdomen vs. peso

[17]:
```
# Selección de características
X_2 = Top[['CirAbdomen']].values
y = Top[['Peso']].values

# Selección de columna
X_col_2 = Top.columns[1:2]
# Aplicación y visualización del modelo
fun.AARLS(X_2,y)
fun.AATrainDisperGraf(X_col_2, y_col, X_2, y)
fun.AATestDisperGraf(X_col_2,y_col,X_2,y)
```

Estadísticos de la regresión

```
Valor del intercepto: -20.82.
Valor del coeficiente: 1.1.
Primeros cinco valores de la regresión: [61.33, 85.66, 89.41, 84.12, 118.15].
Coeficiente de Determinación: 77.7%.
```

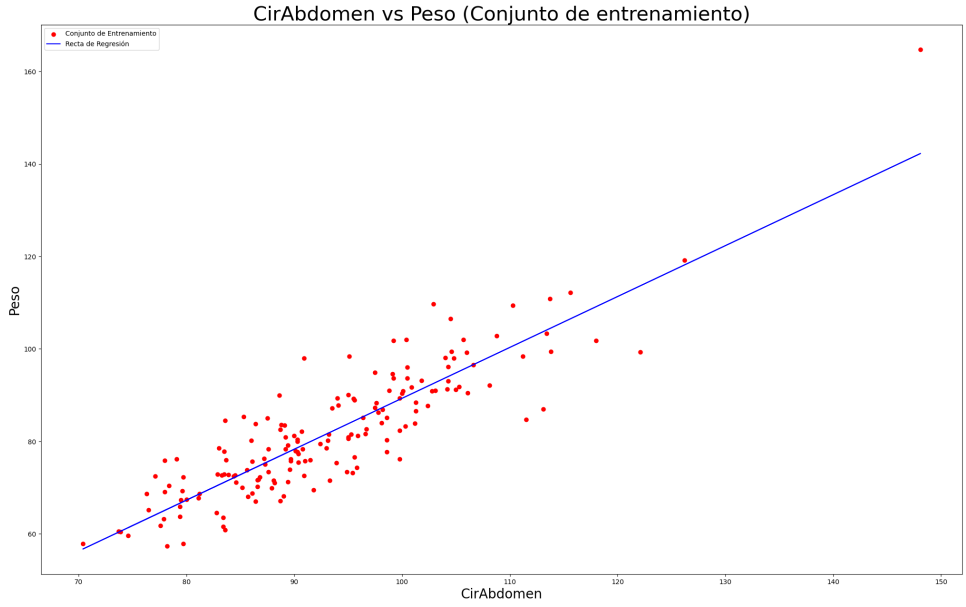

Figura 3.11: Dispersión de los valores de entrenamiento (en rojo) contrastada con la recta de regresión generada (en azul).

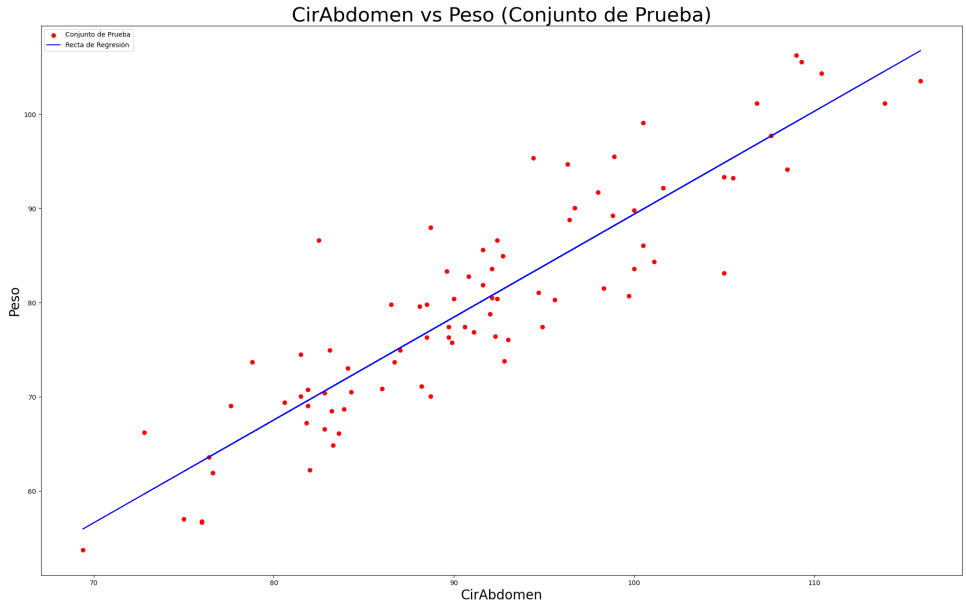

Figura 3.12: Dispersión de los valores de prueba (en rojo) contrastada con la recta de regresión generada (en azul).

El modelo de regresión lineal simple paramétrico (Circunferencia de abdomen vs Peso), utilizando aprendizaje automático, tiene entonces la siguiente forma:

$$y = -20.82 + 1.1x.$$

Y, el modelo de estimación, tiene la siguiente forma:

$$\hat{y}_i = -20.82 + 1.1x_i.$$

Circunferencia de muslo vs. peso

[19]:
```
# Selección de características
X_3 = Top[['CirMuslo']].values
y = Top[['Peso']].values

# Selección de columna
X_col_3 = Top.columns[2:3]

# Aplicación y visualización del modelo
fun.AARLS(X_3,y)
fun.AATrainDisperGraf(X_col_3, y_col, X_3, y)
fun.AATestDisperGraf(X_col_3,y_col,X_3,y)
```

Estadísticos de la regresión

```
Valor del intercepto: -52.91.
Valor del coeficiente: 2.27.
Primeros cinco valores de la regresión: [64.41, 81.65, 88, 77.34, 111.6].
Coeficiente de Determinación: 78.95%.
```

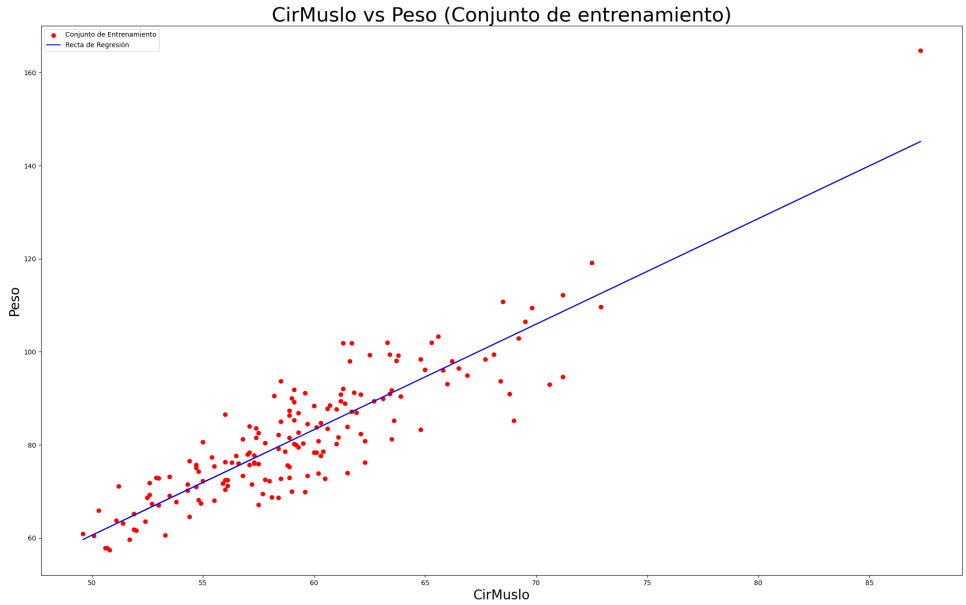

Figura 3.13: Dispersión de los valores de entrenamiento (en rojo) contrastada con la recta de regresión generada (en azul).

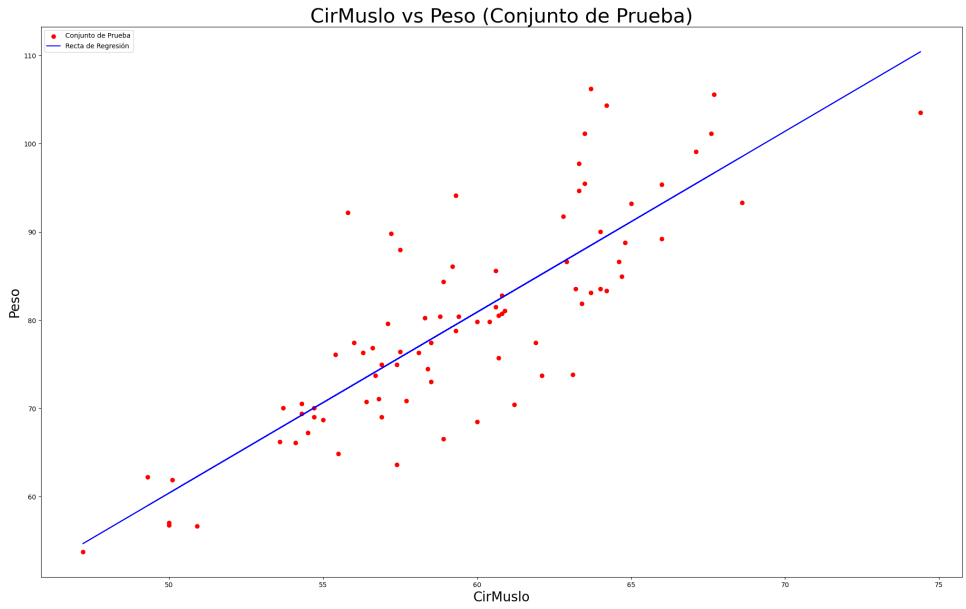

Figura 3.14: Dispersión de los valores de prueba (en rojo) contrastada con la recta de regresión generada (en azul).

El modelo de regresión lineal simple paramétrico (Circunferencia de muslo vs Peso), utilizando aprendizaje automático, tiene entonces la siguiente forma:

$$y = -52.91 + 2.27x.$$

Y, el modelo de estimación, tiene la siguiente forma:

$$\hat{y}_i = -52.91 + 2.27x_i.$$

Despliegue

Al comparar los resultados de ambas metodologías, se nota que los valores de las β's son los mismos. Calculando el error cuadrádo medio de ambas metodologías se obtiene lo siguiente:

[21]:
```
fun.ECM_RLS(X,y) # Función del catálogo para calcular el ECM
```

```
============================================================
ECM Aprendizaje Automático:  21.158443893250226
------------------------------------------------------------
ECM Máxima Verosimilitud:  21.158443893250226
============================================================
Diferencia cuadráda:  0.0
============================================================
```

Tabla 3.3: Tabla de Resumen con el Error Cuadrático Medio

Como se muestra en la Tabla 3.3 el cálculo del error cuadrático medio resulta ser prácticamente el mismo en ambas metodologías. De modo tal que distan entre sí en 1.262×10^{-29}, lo cual es de bastante cercano al cero.

De modo que no existe una diferencia significante entre ambas metodologías, pero es más recomendable el uso de paqueterías por la capacidad de almacenamiento de variables.

3.5.2. Regresión Lineal Múltiple

Se utiliza la base de datos anterior para el modelo de regresión lineal múltiple. Se hace uso de todas las características de la base, exceptuando la circunferencia del pecho.

Como la variable de interés es el 'Peso' se procede a mover toda la columna a la última posición de la base para facilitar la posterior selección de variables.

[1]:
```
pc = datos.pop('Peso')
datos.insert(14, 'Peso', pc)
```

Exploración

Se muestra una matriz de gráficos contrastando los valores entre sí:

[2]:
```
X_col = data.columns[:-1]
y_col = data.columns[-1:]

fun.MatrizDiagramasDispersion(5,3,X_col,y_col,data)
```

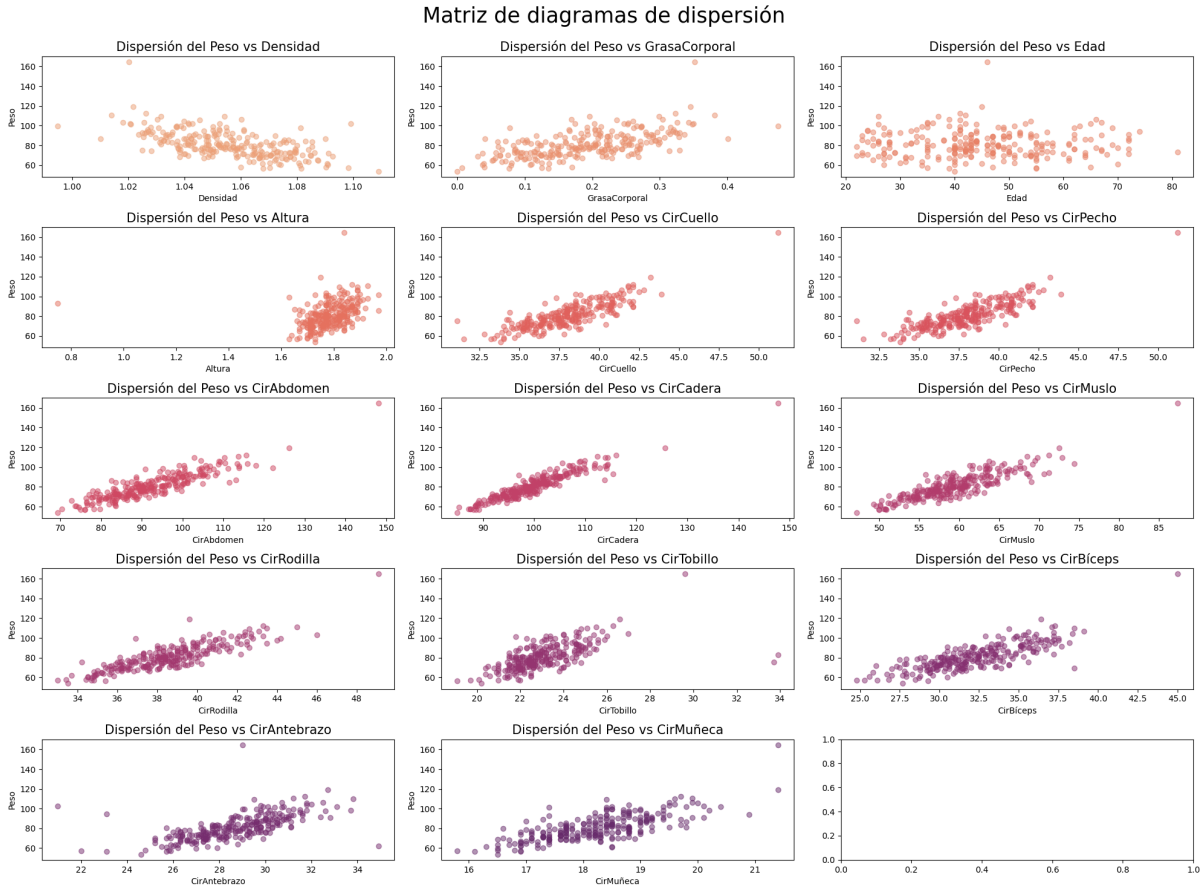

Figura 3.15: Matriz de Gráficos de Dispersión.

En la Figura 3.15 el eje y de todas las gráficas de dispersión muestra el peso contrastado con los valores de las demás características en el eje x.

Existe una relación lineal notoria del peso con cada una de las variables restantes a excepción de los valores de densidad, edad y altura. Con esto se tiene un primer acercamiento del comportamiento de los datos y de los valores con mayor importancia para el modelo.

De igual manera se nota que las variables que corresponden a la circunferencia del cuello y la circunferencia del pecho son prácticamente idénticas. Esto refuerza lo obtenido en la matriz de correlación de la Figura 3.15 respecto a que ambos conjuntos de valores son los mismos. De modo que se opta por eliminar la circunferencia de cuello.

[27]:
```
datos = datos.drop("CirCuello", axis = 1)
```

Implementación del Modelo

Mínimos Cuadrados

Selección de Variables

Teniendo ya una mejor idea para la construcción del modelo, se selecciona la variable dependiente (Y) y las variables independientes (x_i) como lo son la densidad, grasa corporal, edad, altura, circunferencia del cuello, circunferencia del pecho, circunferencia del abdomen, circunferencia de la cadera, circunferencia del muslo, circunferencia de rodilla y la circunferencia del tobillo.

[28]:
```
X = datos.iloc[:,:-1].values
y = datos.iloc[:,-1:].values
```

[5]:
```
fun.MinCua(X,y)
```

Estadísticos de la regresión

```
Matriz de valores de beta:
[-114.19, -32.35, -0.06, 11.59, 0.68, 0.39, 0.67, 0.06, 0.6, 0.36, 0.21, 0.2]

Primeros 10 valores de la regresión:
[57.92, 84.67, 92.67, 82.88, 120.35, 97.13, 80.99, 89.48, 67.54]
```

El modelo de regresión lineal múltiple paramétrico, utilizando mínimos cuadrados, tiene la siguiente forma:

$$y = \mathbb{X} \begin{bmatrix} -114.19 \\ -32.35 \\ -0.06 \\ 11.59 \\ 0.68 \\ 0.39 \\ 0.67 \\ 0.06 \\ 0.6 \\ 0.36 \\ 0.21 \\ 0.2 \end{bmatrix} = \mathbb{X}\hat{\beta}.$$

Y, el modelo de estimación, tiene la siguiente forma:

$$\hat{y}_i = \mathbb{X} \begin{bmatrix} -114.19 \\ -32.35 \\ -0.06 \\ 11.59 \\ 0.68 \\ 0.39 \\ 0.67 \\ 0.06 \\ 0.6 \\ 0.36 \\ 0.21 \\ 0.2 \end{bmatrix} = \mathbb{X}\hat{\beta}.$$

Aprendizaje Automático

[6]:
```
fun.AARLM(X,y)
```

```
Matriz de valores de beta:
[101.88, 15.63, -0.06, 11.43, 0.66, 0.41, 0.64, 0.08, 0.59, 0.4, 0.25, 0.18]

Primeros 10 valores de la regresión:
[58.21, 84.45, 92.59, 82.76, 120.09, 97.06, 80.95, 89.35, 67.58]
```

El modelo de regresión lineal múltiple paramétrico, utilizando aprendizaje automático, tiene la siguiente forma:

$$y = \mathbb{X} \begin{bmatrix} 101.88 \\ 15.63 \\ -0.06 \\ 11.43 \\ 0.68 \\ 0.39 \\ 0.67 \\ 0.06 \\ 0.6 \\ 0.36 \\ 0.21 \\ 0.2 \end{bmatrix} = \mathbb{X}\hat{\beta}.$$

Y, el modelo de estimación, tiene la siguiente forma:

$$\hat{y}_i = \mathbb{X} \begin{bmatrix} -114.19 \\ -32.35 \\ -0.06 \\ 11.59 \\ 0.68 \\ 0.39 \\ 0.67 \\ 0.06 \\ 0.6 \\ 0.36 \\ 0.21 \\ 0.2 \end{bmatrix} = \mathbb{X}\hat{\beta}.$$

En la Tabla 3.4 se muestran a detalle los estadísticos del modelo de regresión lineal múltiple hecho con aprendizaje automático; además de una tabla que muestra el desglose de cada valor de `x` con estadísticos de interés.

Despliegue

Calculando el ECM para ambas metodologías se tiene que:

[8]:
```
fun.ECM_RLM(X,y)
```

```
------------------------------------------------------------
ECM Aprendizaje Automático: 6.91
------------------------------------------------------------
ECM Mínimos Cuadrádos: 7.43
------------------------------------------------------------


------------------------------------------------------------
Diferencia cuadráda: 0.27
------------------------------------------------------------
```

Tabla resumen de la regresión:

OLS Regression Results

Dep. Variable:	y	R-squared:	0.964
Model:	OLS	Adj. R-squared:	0.961
Method:	Least Squares	F-statistic:	318.1
Prob (F-statistic):	4.69e-104	Log-Likelihood:	-400.75
No. Observations:	168	AIC:	829.5
Df Residuals:	154	BIC:	873.2
Df Model:	13	Covariance Type:	nonrobust

	coef	std err	t	P>|t|	[0.025	0.975]
const	-234.8248	68.888	-3.409	0.001	-370.912	-98.738
x1	101.8809	63.618	1.601	0.111	-23.795	227.557
x2	15.6282	14.898	1.049	0.296	-13.802	45.059
x3	-0.0647	0.027	-2.441	0.016	-0.117	-0.012
x4	11.4322	2.344	4.878	0.000	6.802	16.062
x5	0.6578	0.171	3.843	0.000	0.320	0.996
x6	0.4063	0.072	5.609	0.000	0.263	0.549
x7	0.6418	0.095	6.730	0.000	0.453	0.830
x8	0.0785	0.116	0.674	0.501	-0.152	0.309
x9	0.5937	0.176	3.379	0.001	0.247	0.941
x10	0.3982	0.150	2.648	0.009	0.101	0.695
x11	0.2488	0.137	1.819	0.071	-0.021	0.519
x12	0.1817	0.168	1.079	0.282	-0.151	0.514
x13	0.6191	0.430	1.439	0.152	-0.231	1.469

Tabla 3.4: Tabla resumen de la regresión múltiple

En el despliegue se muestra que el ejemplo en donde se utilizó aprendizaje automático tiene un error cuadrado medio menor en 0.5214 en contraste con la metodología de mínimos cuadrados. Esto sucede por el almacenamiento de decimales. La distancia que existe entre estas estimaciones del error, aunque menor a uno, es notable.

De modo que la estimación de regresión lineal múltiple es más acertada utilizando aprendizaje automático.

3.5.3. Regresión Logística

Como la estimación de los parámetros es numérica, sólo se aborda el enfoque de aprendizaje automático.

Se cuenta con una base de datos con 918 personas de las cuales se conocen las siguientes características:

- Edad.
- Sexo.
- ChestPainType: tipo de dolor torácico. (TA: angina típica, ATA: angina atípica, NAP: dolor no anginoso, ASY: asintomático).
- Presión arterial en reposo.
- Colesterol - BS en ayunas: azúcar en sangre en ayunas. (1: si BS en ayunas $>$ 120 mg/dl, 0: de otra forma).
- ECG en reposo: resultados del electrocardiograma en reposo. (Normal: normal, ST: con anomalías en la onda ST-T (inversión de la onda T y/o elevación o depresión del ST $>$ 0,05 mV), HVI: hipertrofia ventricular izquierda probable o definitiva según los criterios de Estes).
- MaxHR: frecuencia cardíaca máxima alcanzada. (Valor numérico entre 60 y 202).
- ExerciseAngina: angina inducida por el ejercicio. (Y: Sí, N: No).
- Oldpeak: ST. (Valor numérico medido en depresión).
- ST_Slope: Pendiente del segmento ST del ejercicio máximo. (Arriba: pendiente ascendente, Plana: plana, Abajo: pendiente descendente).
- Enfermedad cardíaca: clase de salida. (1: enfermedad cardíaca, 0: normal).

Dicha base de datos fue extraida de la página kaggle.com.
La meta es pronosticar si un paciente con estas características podría o no desarrollar una enfermedad cardíaca en función de los valores que dicho paciente presente en cada característica.

Bibliotecas

[1]:
```
import pandas as pd # Paquetería para cargar archivos
import Catalogo_de_Funciones as fun # Paquetería del catálogo de funciones
```

Exploración

[2]:
```
path = 'heart.csv'
fun.head_describe_csv(path)
```

Índice	ID	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS
0	313	40	M	ATA	140	289	0
1	162	49	F	NAP	160	180	0
2	216	37	M	ATA	130	283	0
3	293	48	F	ASY	138	214	0
4	255	54	M	NAP	150	195	0

Índice	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	Normal	172	N	0	Up	0
1	Normal	156	N	1	Flat	1
2	ST	98	N	0	Up	0
3	Normal	108	Y	1.5	Flat	1
4	Normal	122	N	0	Up	0

Tabla 3.5: Primeros cinco individuos con sus valores.

Estadísticos	Age	RestingBP	Cholesterol	FastingBS
count	918	918	918	918
mean	53.510893	132.396514	198.799564	0.233115
std	9.432617	18.514154	109.384145	0.423046
min	28	0	0	0
25 %	47	120	173.25	0
50 %	54	130	223	0
75 %	60	140	267	0
max	77	200	603	1

Estadísticos	MaxHR	Oldpeak	HeartDisease
count	918	918	918
mean	136.809368	0.887364	0.553377
std	25.460334	1.06657	0.497414
min	60	-2.6	0
25 %	120	0	0
50 %	138	0.6	1
75 %	156	1.5	1
max	202	6.2	1

Tabla 3.6: Descripción de los datos.

La Tabla 3.5 muestra una extracción de los primeros cinco individuos de la base de datos con sus respectivos valores en cada característica. Además, la Tabla 3.6 muestra estadísticos de importancia de la base de datos.

Preparación de los Datos

Se comienza por eliminar los valores nulos de la base de datos para posteriormente quitar las columnas que no son de importancia para el modelo que se desea realizar. En este ejemplo la columna de *ID* no aporta al modelo.

[3]:
```
datos.dropna(inplace=True)
datos.drop('ID', axis=1, inplace=True)
```

Matriz de Correlación

[4]:
```
data = pd.read_csv('heart.csv')
data.dropna(inplace=True) # Limpieza de nulos
data.drop('ID', axis=1, inplace=True) # Eliminación de la columna de ID
fun.MatrizCorrelación(data)
```

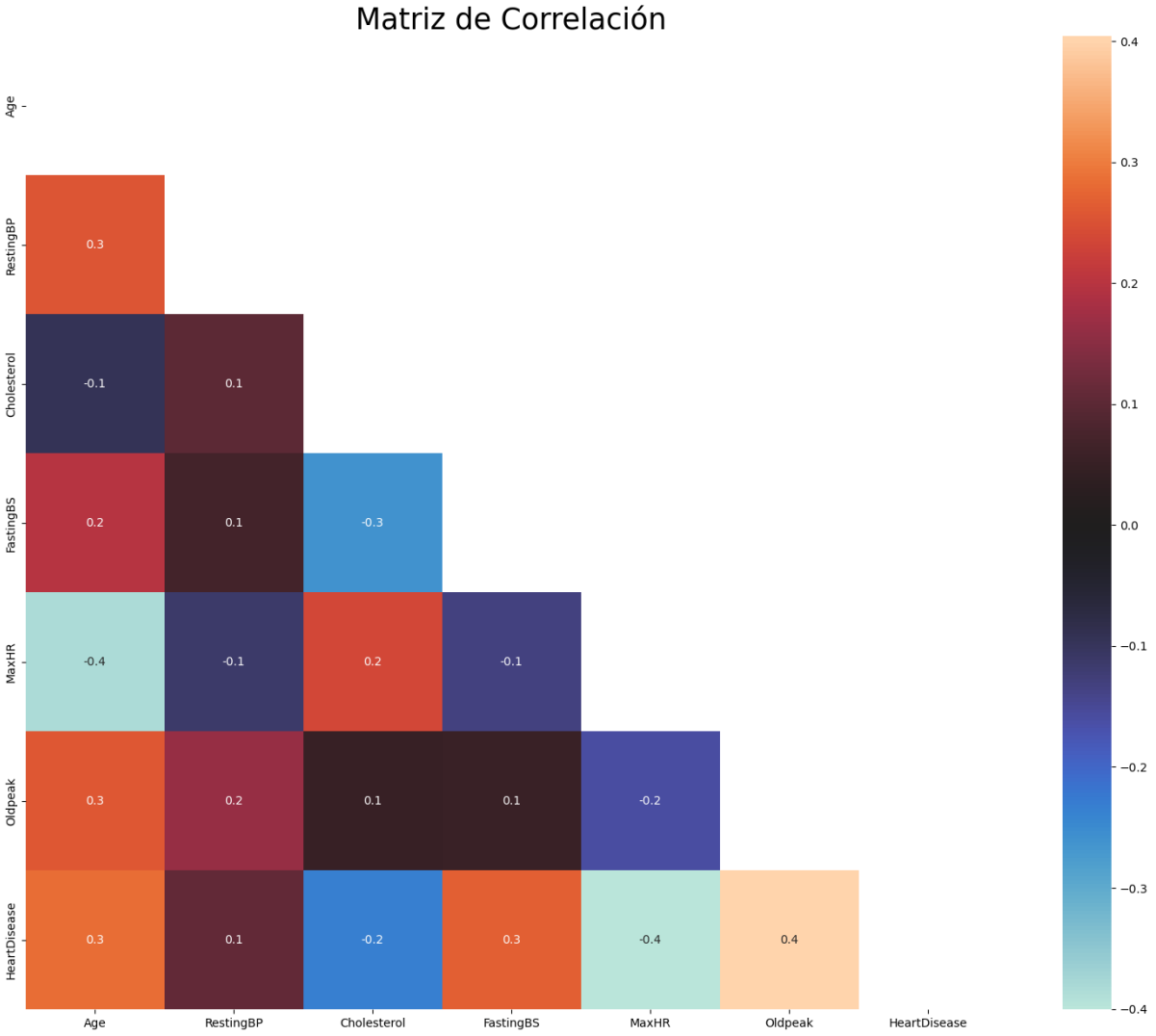

Figura 3.16: Matriz de Correlación con los Valores Iniciales

Dentro de los datos no categóricos se puede notar que no existe una correlación fuerte entre las variables de la base, como se muestra en la Figura 3.16. Por consiguiente se convierten las variables categóricas en *dummy* y se recalcula la matriz de correlación.

Nueva Matríz de Correlación

[5]:
```
fun.MatrizCorrelación(data)
```

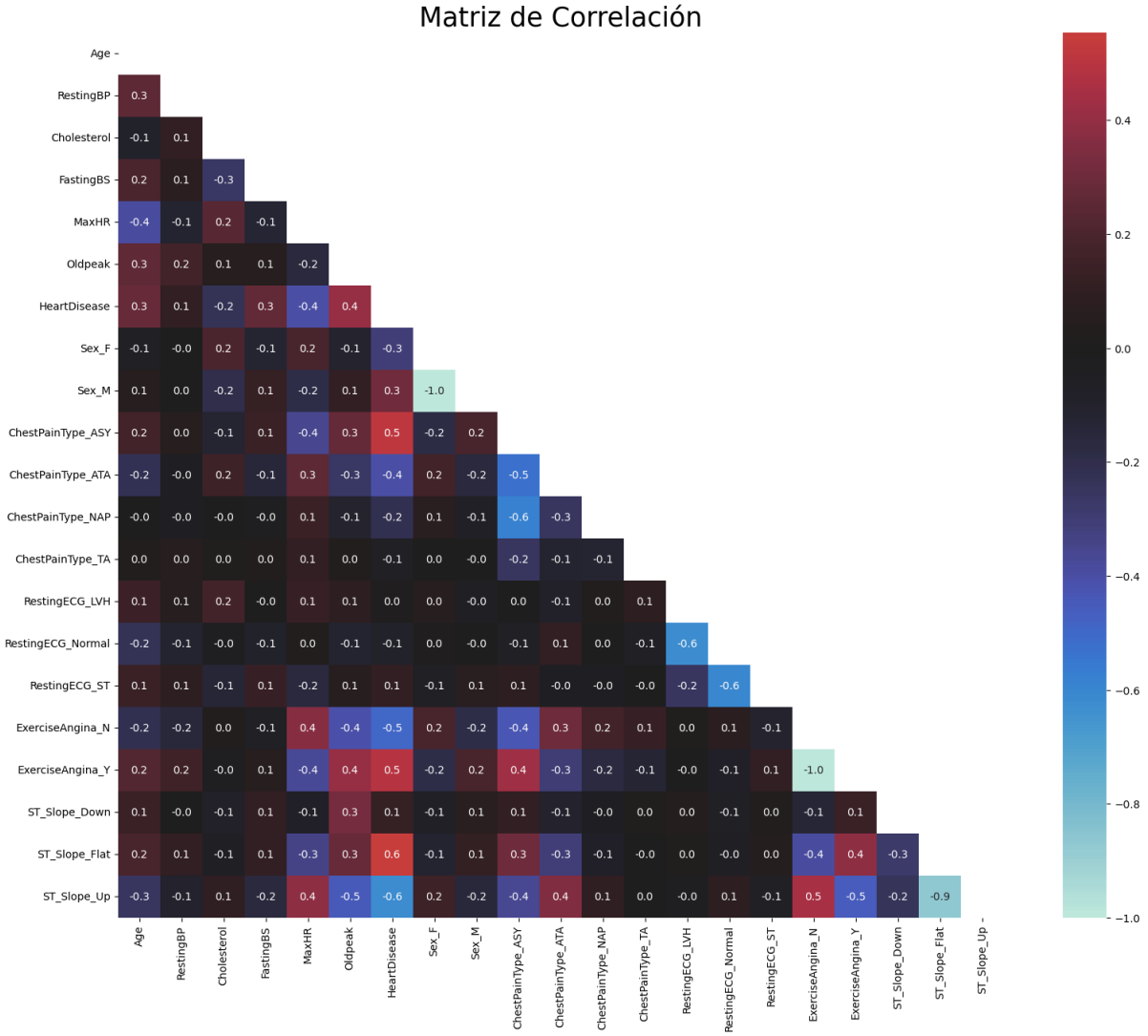

Figura 3.17: Matriz de Correlación de Variables Tipo Dummy.

Al tener esta nueva matriz de correlación (Figura 3.17) se observa que la variable de *HeartDisease* es la que se encuentra mejor correlacionada con las demás variables. Por consiguiente se vuelve un excelente candidato para realizar la estimación y pronóstico sobre ella.

A continuación se mueve de lugar en la base de datos la columna de interés por conveniencia.

[6]:
```
pc = datos.pop('HeartDisease')
datos.insert(20, 'HeartDisease', pc)
```

Visualización de los Valores

[7]:
```
pc = data.pop('HeartDisease')
data.insert(20, 'HeartDisease', pc)
```

[8]:
```
# Selección de nombres de columnas
X_col = data.columns[:-1]
y_col = data.columns[-1:]

# Visualización de diagramas de dispersión
fun.MatrizDiagramasDispersion(4,5,X_col,y_col,data,(45,40))
```

Figura 3.18: Matriz de gráficos de dispersión.

Implementación del Modelo

Selección de Variables

[9]:
```
X = datos.drop('HeartDisease', axis =1).values
y = datos[['HeartDisease']].values
```

[10]:
```
fun.RegLog(X,y)
```

```
Valores de la regresión:
[1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1.
0. 0. 1. 1. 1. 0. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 0. 1. 1. 0. 1. 0. 0. 1.
1. 0. 1. 0. 0. 1. 1. 1. 1. 0. 0. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1.
0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 1. 1. 0. 1. 1. 0. 0. 1. 0.
1. 0. 0. 0. 1. 0. 1. 1. 1. 1. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 0. 0. 0. 0. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 0. 1. 0. 1.
1. 0. 1. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 1. 1. 0. 1. 1. 1. 0.
1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 0. 0. 1.
1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1.
1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 0. 1.]
```

La matriz de datos anterior muestra los valores pronosticados mediante los valores de prueba. El valor de 1 muestra que el individuo puede que tenga, o desarrolle, un desorden cardíaco; al contrario del valor 0 que indica una posible ausencia de un desorden cardíaco.

Despliegue

Evaluación del Rendimiento del Modelo

Matriz de Confusión

La construcción de una matriz de confusión es necesaria para conocer el total de aciertos que se obtuvo al implementar el modelo y saber el rendimiento obtenido. Esta se realiza mediante los valores de falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos.

Para ello se llama a la función `matrx_conf()` del catálogo de funciones; además de importar el clasificador `LogisticRegression` de la paquetería de `sklearn.linear_model`. Esta tiene por argumentos la matriz de valores de X, los valores de y, el clasificador que se utilizó (regresión logística en este caso), el tamaño del conjunto de entrenamiento, la aleatoriedad (randon state) y el tamaño (en sentido de imágen) de la matriz.

[11]:
```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0)

fun.matrx_conf(X,y,classifier,1/4,0,(7, 5))
```

```
Total de Aciertos: 199.
Total de Fracasos: 31.
Proporción de Aciertos: 86.52%.
```

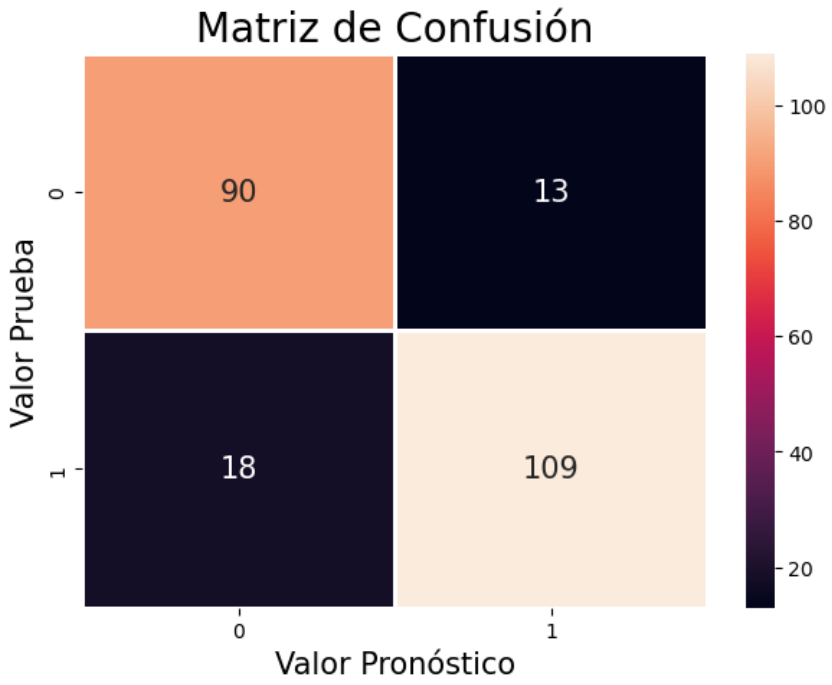

Figura 3.19: Matriz de confusión.

La Figura 3.19 muestra la matriz de confusión asociada al problema; la cual indica un 86.52 % de aciertos utilizando el modelo implementado.

Curva de ROC

La construcción de la curva de ROC, utilizando la paquetería *sklearn*, se realiza calculando las probabilidades del conjunto de prueba (X_test) con el modelo que se calculó.

Para ello se utiliza la función `ROC` del catálogo de funciones para calcular *la tasa de falsos positivos, la tasa de verdaderos positivos* y los *umbrales* correspondientes.

Además, como se puede notar, utiliza los mismos argumentos que la función para construir la matriz de confusión. Dando entonces la siguiente figura:

[12]:
```
fun.ROC(X,y,classifier,1/4,0,(20, 15))
```

```
Sin entrenar: ROC AUC=0.5.
Regresión Logística: ROC AUC=0.946.
```

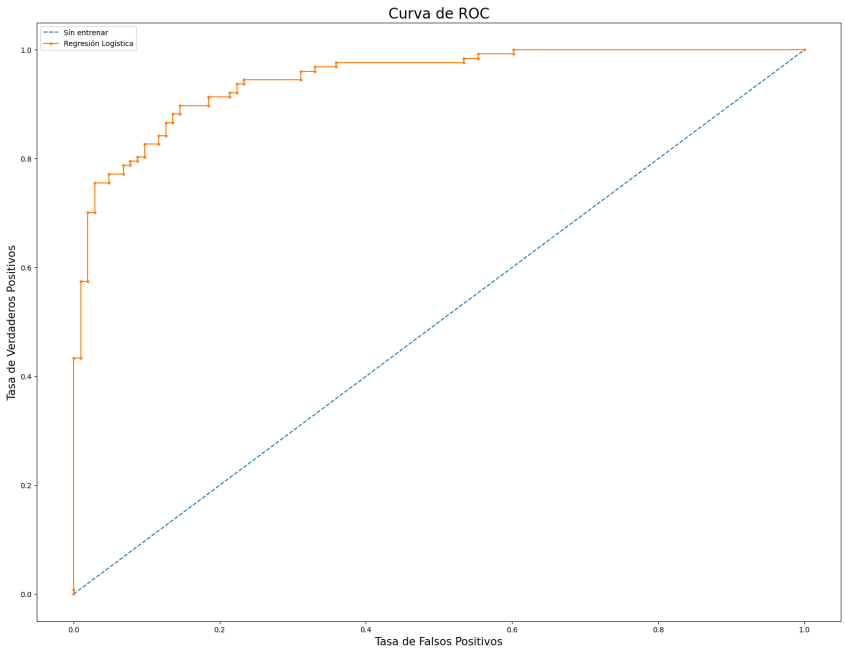

Figura 3.20: Curva de ROC.

Como se muestra en la Figura 3.20 la implementación de un modelo de regresión logística denota un 94.6 % de efectividad al momento de clasificar los valores correctamente.

Conclusión

En la matriz de confusión se ha alcanzando una predicción bastante aceptable con respecto a los datos que el modelo de estimación ha generado. Siendo este acertado en un aproximado de 86.52 %.

La curva de ROC, por su parte, indica que los valores entre 0 y 0.4 de la TFP se encuentra la mejor estimación del modelo; dando entonces que la TVP más óptima se encuentra entre los valores de 0.8 y 0.9. Lo cual se refuerza por el hecho de obtener un *área bajo la curva de ROC* (AUC) de 0.946 (AUC=0.946); que se traduce en una precisión del 94.6 %.

De aquí la pregunta inmediata es si ¿es el mejor modelo para implementar en la base de datos? la verdad es que no lo es y esto lo podemos conocer mediante los *métodos de evaluación de rendimiento* comparando distintas implementaciones entre sí para discernir al modelo que mejor se ajuste. En esto las curvas de ROC juegan un papel importante; ya que se pueden superponer entre sí y, a su vez, evaluar las áreas denotadas por cada curva. Por esto mismo es común que, dentro de un mismo espacio de ROC, habiten distintas curvas generadas a partir de implementaciones diferentes.

El defecto es una virtud mal mirada,
Una virtud mal vista.

Guillermo del Toro, De Geometría a La Forma del Agua.

4

k-Vecino más Cercano

El *k-vecino más cercano* (k-Nearest Neighbor, k-NN por sus siglas en inglés) es un método de aprendizaje supervisado que utiliza la distancia entre los valores de la base de datos. El algoritmo selecciona las k observaciones más cercanas a los nuevos registros generados mediante la predicción. Así, se forman conjuntos de puntos que siguen reglas de distancia específicas o se agrupan en torno a las instancias que se desean predecir. En regresión, el valor de respuesta se obtiene promediando las k observaciones más cercanas. En clasificación, se asigna la clase más común entre esas observaciones [19].

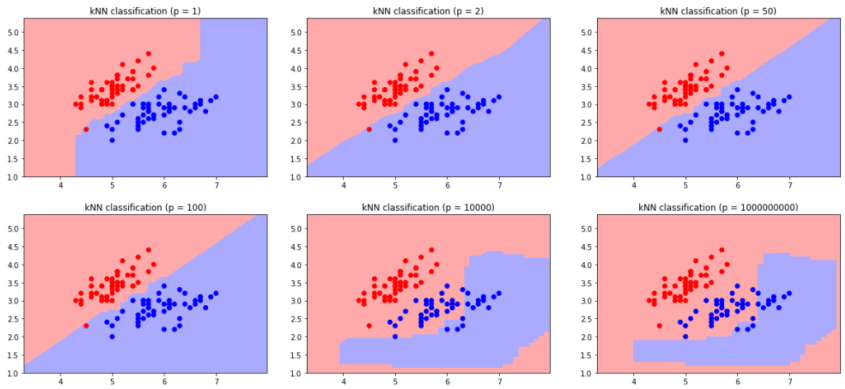

Figura 4.1: Matriz de gráficos que muestra el comportamiento del modelo al escoger diferentes números de vecinos (p en este caso).

En la Figura 4.1 muestra el comportamiento de clasificación utilizando el método de k-NN con diferentes valores de vecinos. Un valor de vecinos muy grande tiende a generar inconsistencias al momento de clasificar, mientras que un valor pequeño es más confiable para la clasificación de valores.

La manera de aplicar una medida de distancia depende de la naturaleza del problema. Por ello se debe tener muy presente el concepto geométrico de **distancia**.

4.1. Distancia

Definición 15 (Distancia)
Dado un conjunto $\mathcal{S} \neq \emptyset$, una distancia sobre $\mathcal{S}$ es una función $\boldsymbol{d}$: $\mathcal{S} \times \mathcal{S} \longmapsto \mathbb{R}$ la cual cumple, $\forall\, x, y, z \in \mathcal{S}$, las siguientes condiciones:

1. $0 \leq d(x, y)$.
2. $d(x, y) = 0 \;\leftrightarrow\; x = y$.
3. *Simetría:* $d(x, y) = d(y, x)$.
4. *Desigualdad Triangular:* $d(x, y) \leq d(x, z) + d(z, y)$.

A la pareja $|S, d|$ se le denomina *espacio métrico*[23].

Con esto es fácil mostrar que las siguientes funciones son, en efecto, distancias:

1. Distancia Euclidiana [23] [24]:

   Dados dos puntos $x, y \in \mathcal{S} \subset \mathbb{R}^2$, donde $x = (x_1, y_1)$ y $y = (x_2, y_2)$, se tiene:

   $$\mathbf{d}_e\,(x, y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

2. Geometría del Taxi (Distancia Manhattan) [16] [24]:

   Dados dos puntos $x, y \in \mathcal{S} \subset \mathbb{R}^2$ donde $x = (x_1, y_1)$ y $y = (x_2, y_2)$, se tiene:

   $$\mathbf{d}_t\,(x, y) = \mid x_1 - x_2 \mid + \mid y_1 - y_2 \mid . \tag{4.1}$$

3. Distancia Mahalanobis [17] [21] donde $x = (x_1, y_1)$ y $y = (x_1, y_2)$ realizaciones de las variables aleatorias X, Y, se tiene:

   Sean dos variables aleatorias X, Y con la misma distribución de probabilidad; $\vec{x}$ y $\vec{y}$ vectores asociados a las variables aleatorias. La distancia de Mahalanobis se define como:

   $$\mathbf{d}_m\,(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})} = \sqrt{\left(\frac{x_1 - y_1}{\sigma_1}\right)^2 + \left(\frac{x_2 - y_2}{\sigma_2}\right)^2}. \tag{4.2}$$

   donde σ_1 y σ_2 representan la varianza de los vectores de medida correspondientes.

4.2. Similitud

Cuando se menciona la similitud dentro del modelo de k-NN, habla de qué tan parecidos, o similares, son los vecinos entre sí [25]. Para *medir la similitud* de las observaciones: Sea k el número de vecinos:

- $k = 1$, habla de la versión más sencilla del algoritmo ya que sólo se considera un vecino.
- $k > 1$, habla de similitudes entre varios individuos cercanos entre sí, tomando en cuenta como está siendo definida la distancia para el modelo.

  Para conocer cuando es óptimo tomar un k de esa manera se hace una comparación entre los modelos predictivos que se generan con el conjunto de prueba y entrenamiento. Existen métodos los cuales ayudan en la búsqueda del número de vecinos más óptimo. Los más usuales son utilizando gráficos que comparen el puntaje obtenido en determinados valores de k[1] y mediante las *Curvas de ROC*.

[1] Se comenta esta parte más a detalle en el ejemplo con Python.

4.3. Ejemplo con Python

Esta base de datos proviene del Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales [1] [7]. El objetivo es predecir si un paciente presenta diabetes dadas las demás características extraídas de los individuos. La muestra de 768 pacientes está compuesta por mujeres de al menos 21 años con herencia genética del pueblo indígena pima [8]. Las características son:

- TotEmbarazos: Número total de veces que la persona estuvo en etapa gestante.
- Glucosa: concentración de glucosa en plasma a las 2 horas en una prueba de tolerancia oral a la glucosa.
- Presión arterial: presión arterial diastólica (mm Hg).
- Grosor de la piel: Grosor del pliegue cutáneo del tríceps (mm).
- Insulina: insulina sérica de 2 horas (mu U/ml).
- IMC: Índice de masa corporal (peso en $kg/(AlturaEnM)^2$).
- DiabetesPedigreeFunction: función de pedigrí de diabetes.
- Edad: Edad (años).
- Resultado: variable de clase (0 o 1). La cual denota si el individuo vive con diabetes (1) o no (0).

Bibliotecas

[2]:
```
import pandas as pd # Paquetería para cargar archivos
import Catalogo_de_Funciones as fun # Paquetería del catálogo de funciones

from sklearn.preprocessing import StandardScaler # Escalar valores
```

Se comienza por extraer pocos valores de la base de datos para conocer como está constituida y generar estadísticos globales para un primer acercamiento de los valores que formarán parte del modelo.

[3]:
```
path = 'diabetes.csv'
fun.head_describe_csv(path)
```

Índice	TotEmbarazos	Glucosa	PresiónArterial	GrosorPiel	Insulina
1	6	148	72	35	0
2	1	85	66	29	0
3	8	183	64	0	0
4	1	89	66	23	94
5	0	137	40	35	168

Índice	IMC	FuncionDiabetesPedigree	Edad	Resultado
1	33.6	0.627	50	1
2	26.6	0.351	31	0
3	23.3	0.672	32	1
4	28.1	0.167	21	0
5	43.1	2.288	33	1

Tabla 4.1: Extracción de los valores de los primeros cinco individuos de la base de datos.

Estadísticos	count	mean	std	min	25\ %
TotEmbarazos	768	3.845052	3.369578	0	1
Glucosa	768	120.894531	31.972618	0	99
PresiónArterial	768	69.105469	19.355807	0	62
GrosorPiel	768	20.536458	15.952218	0	0
Insulina	768	79.799479	115.244002	0	0
IMC	768	31.992578	7.88416	0	27.3
FuncionDiabetesPedigree	768	0.471876	0.331329	0.078	0.24375
Edad	768	33.240885	11.760232	21	24
Resultado	768	0.348958	0.476951	0	0

Estadísticos	50\ %	75\ %	max
TotEmbarazos	3	6	17
Glucosa	117	140.25	199
PresiónArterial	72	80	122
GrosorPiel	23	32	99
Insulina	30.5	127.25	846
IMC	32	36.6	67.1
FuncionDiabetesPedigree	0.3725	0.62625	2.42
Edad	29	41	81
Resultado	0	1	1

Tabla 4.2: Tabla con estadísticos generales de la base de datos.

La Tabla 4.1 muestra a cinco individuos de la base de datos con cada valor asignado a la característica correspondiente. Al realizar una extracción de este estilo se comienza a entender (muy por encima) el comportamiento, estructura y distribución de la base. Por ejemplo, se observa que la variable de *Función Diabetes Pedigree*, cuando está por encima de 0.5, se relaciona de manera directa con los individuos que viven con diabetes. Sin embargo, es de notar que existen valores inconsistentes en la base de datos. En particular, en las variables de *Grosor de Piel* y de *Insulina* se tienen valores de cero. Esto indica que dentro de la base existen más inconsistencias.

La Tabla 4.2 muestra una serie de estadísticos de interés con los cuales se determinan si existen datos atípicos en una primera inspección. Como se muestra, las columnas de *Glucosa, PresiónArterial, GrosorPiel, Insulina y IMC* cuentan con valores de cero que carecen de una interpretación real. De modo que pasan a removerse de la base dichos valores de cero.

[4]:
```
data = pd.read_csv(path)

data[['Glucosa','PresiónArterial',
'GrosorPiel','Insulina','IMC']] = data[['Glucosa','PresiónArterial',
'GrosorPiel','Insulina','IMC']].replace(0,np.NaN)

data = data.dropna()
```

Con esto se obtiene una muestra de tamaño 392; lo que es, en proporción, el 51.0416 % de los datos originales. Hecho esto se procede a calcular la matriz de correlación de la siguiente forma:

Matriz de Correlación

[5]:
```
fun.MatrizCorrelación(data)
```

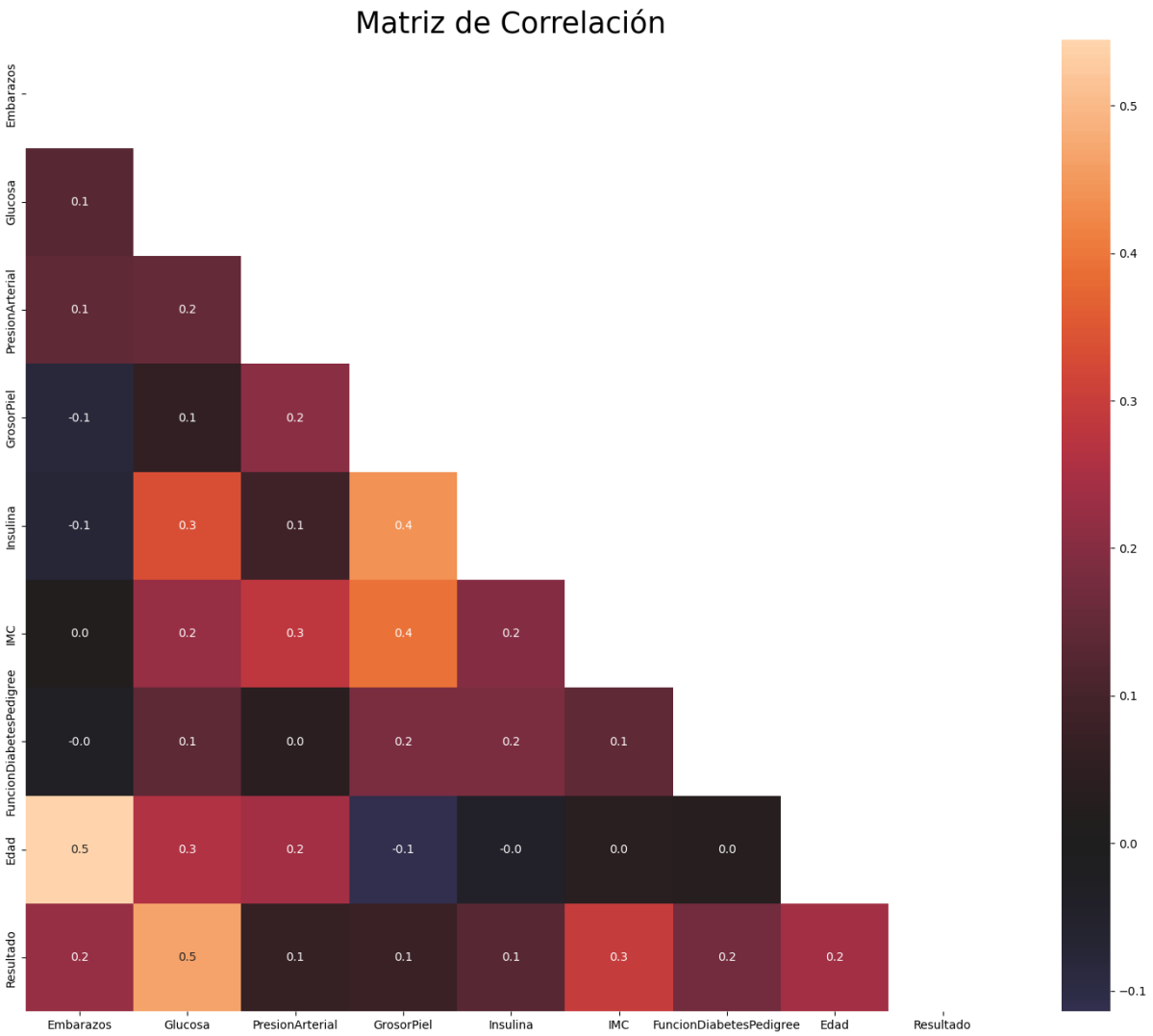

Figura 4.2: Matriz de Correlación.

La Figura 4.2 representa la matriz de correlación de la base de datos. Esta matriz señala, con un color claro, los valores más cercanos a uno entre dos características y, con un color oscuro, los valores más cercanos a cero. De este modo se puede notar que, con un valor de 0.52, la característica que mejor se relaciona con el *Resultado* es la de *Glucosa*. Sin embargo, la glucosa está relacionada (con un valor de 0.56) con la característica de *Insulina*. Esto ayuda a plantear una primera hipótesis donde, los valores que mejor ayudan a la clasificación, son la insulina y la glucosa.

Para conocer mejor la acumulación de valores de cada una de las variables para encontrar inconsistencias y patrones dentro de la base se crean una serie de diagramas de cajas y bigotes como sigue:

Exploración

[7]:
```
X = data.drop("Resultado", axis =1)
y = data.iloc[:,-1:]
fun.MatrizCajasBigotes(4,2,X,data)
```

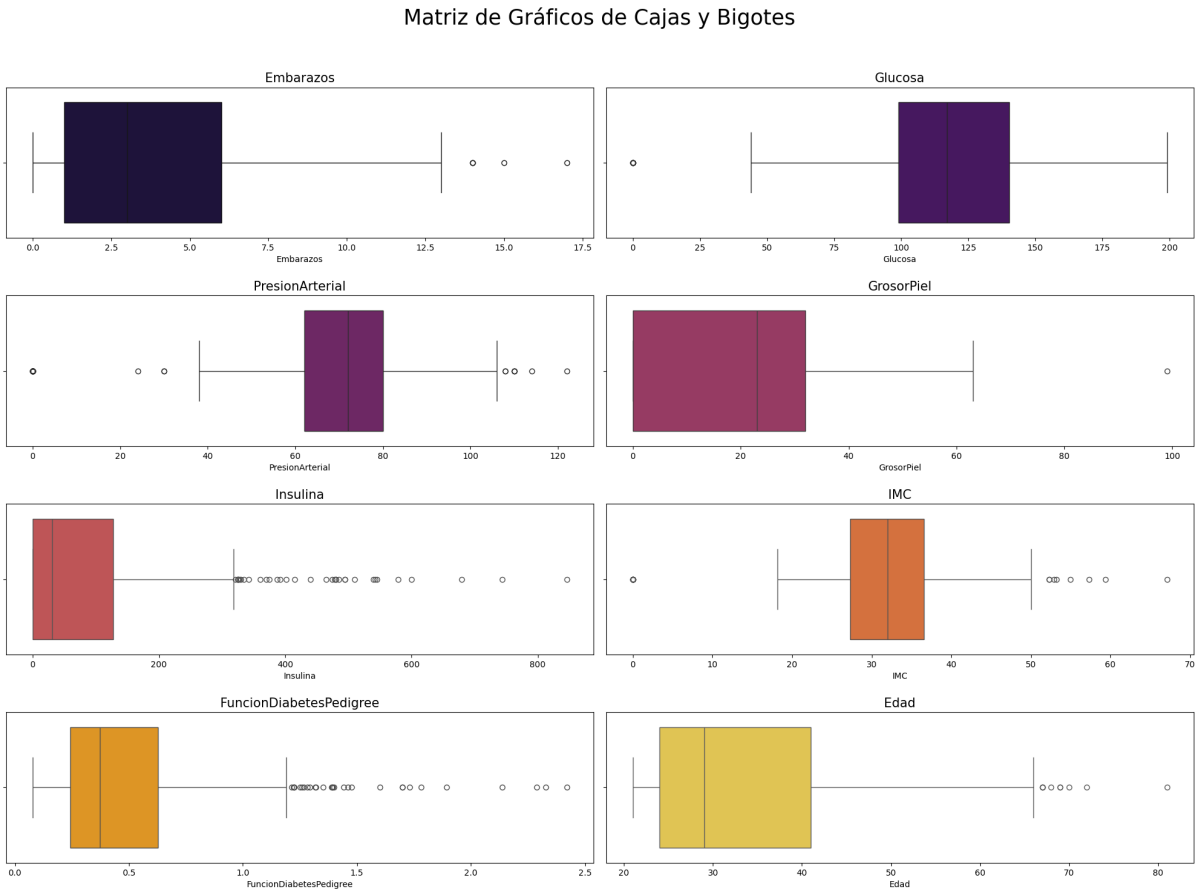

Figura 4.3: Diagrama de Cajas y Bigotes.

Aunque en los diagramas de insulina y función diabétes pedigree de la Figura 4.3 presentan una cantidad notable de datos atípicos, en las demás figuras se observan en una cantidad mucho menor o, en el caso de Glucosa, inexistentes.

De modo que se procede a modelar el problema observando si existe alguna relación entre las valores de las características. Para ello se grafican todos los valores contrastados entre sí; lo cual genera una serie de figuras como sigue:

[8]:
```
fun.DistKDE(data)
```

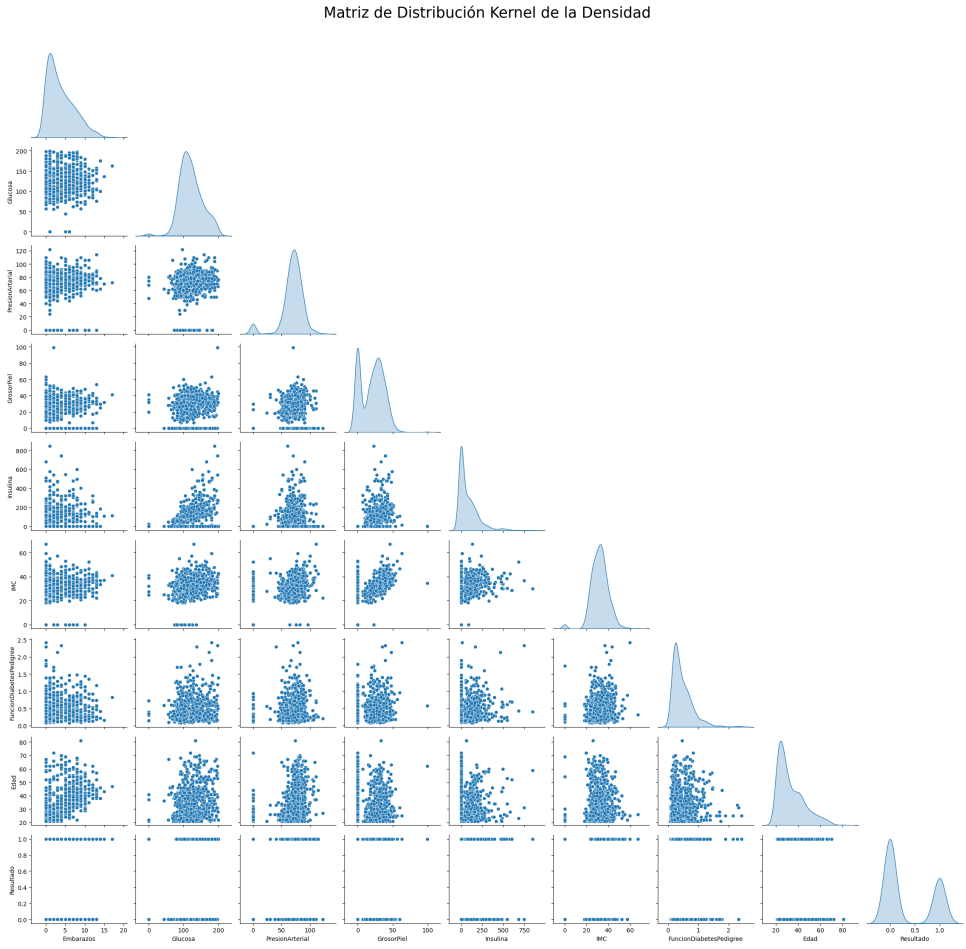

Figura 4.4: Matriz de gráficos de dispersión con la distribución kernel de la densidad.

Las gráficas de la Figura 4.4 en la diagonal principal muestran una *distribución kernel de la densidad* (KDE), por sus siglas en inglés, de cada característica y el resto presentan gráficos de dispersión. En el caso de los valores de la fila de 'Resultados', al ser estos binarios, presentan ese tipo de acumulación. Es de notar que los valores mostrados a la izquierda de la figura anterior no necesariamente tienen relación con las gráficas en la diagonal principal. Como resulta de interés analizar por separado la característica de resultados, se grafica como sigue:

[11]:
```
fun.HistogramaKDE(data,y)
```

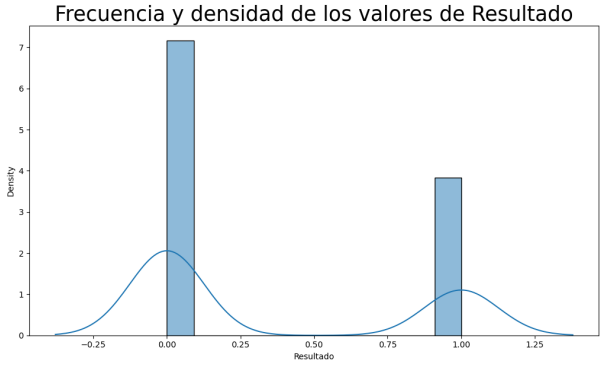

Figura 4.5: Densidad del resultado.

Con esto es fácil visualizar la frecuencia y densidad de los valores de la columna de Resultado. Para comenzar con la implementación del modelo es necesario conocer como se relaciona la variable de *Resultado* contra todas las demás en un sentido de clasificación. Es decir, se grafican los individuos separados en tantos conjuntos como tipos únicos tiene la columna para visualizar una distinción, conglomeración o relación entre estos tipos. Para conocer dicho comportamiento, se grafican los valores de las características haciendo uso de la función `kde_plt_hue()` de la siguiente manera:

[9]:
```
fun.kde_plt_hue(data,y)
```

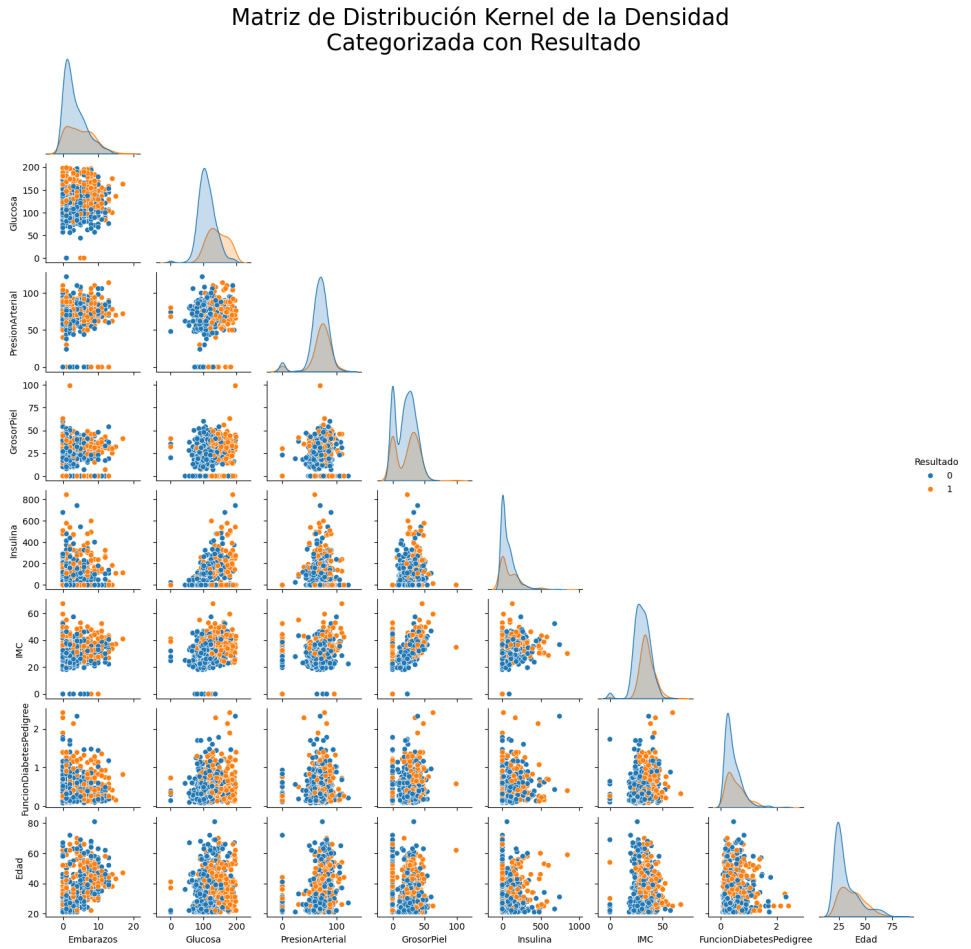

Figura 4.6: Matriz de Gráficos de Dispersión con la Densidad Kernel.

Como los valores de la columna de 'Resultados' se dividen en los tipos 0 y 1 se hacen distinción entre sí con los colores azul y naranja (ver Figura 4.6). A lo largo de la diagonal principal se genera una estimación kernel de la densidad para los valores de cero y uno. Para ejemplificarlo mejor se toma la sección de la figura anterior correspondiente a la Glucosa y el IMC tal que:

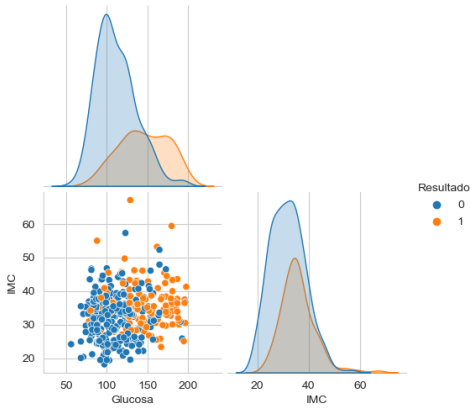

Figura 4.7: IMC y Glucosa Divididos por Resultado.

Se obtienen tres gráficas que indican distintas cosas. En particular se aísla la gráfica de dispersión para su interpretación tal que:

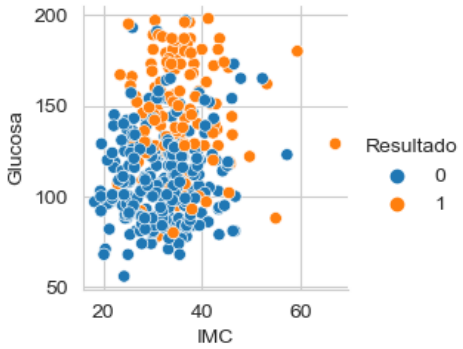

Figura 4.8: Gráfica de Dispersión del IMC y la Glucosa Dividido por Resultado.

La mayor concentración de puntos azules se encuentra por debajo de un puntaje de 40 en IMC y aproximadamente por debajo de 125 de glucosa; al igual que los puntos naranjas. Así entonces se conoce la manera en la cual están acumulados los tipos de la variable de resultado contra, en este caso, los valores de glucosa e IMC y así en general para la Figura 4.6.

Para los gráficos de la diagonal principal se utiliza un código similar al anterior, solo tomando los mismos valores tanto para el argumento 'y_vars' como 'x_vars':

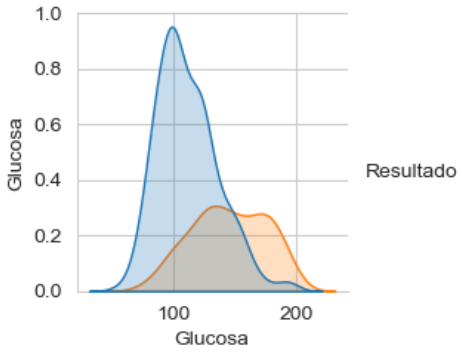

Figura 4.9: Densidad Kernel de la Glucosa Dividido por Resultado.

En la Figura 4.9 el color azul denota los individuos con un valor de cero en la característica de 'Resultado', mientras que el naranja muestra los que tienen un valor de uno. De la figura anterior se pueden interpretar varias cosas; una de ellas es que la media de glucosa de las personas que no viven con diabetes es cercana a 100, mientras que las personas denotadas en color naranja presentan una mayor acumulación por encima de 125 y por debajo de 200 de glucosa. Para ayudar más al análisis gráfico la paquetería *seaborns* adhiere la densidad kernel estimada para los valores de las columnas utilizando la característica de 'Resultado'. Esto para conocer la acumulación de los valores de cero y uno en relación con su acomodo, así que:

```
[10]: fun.kde_plt_hue_cat(data, y)
```

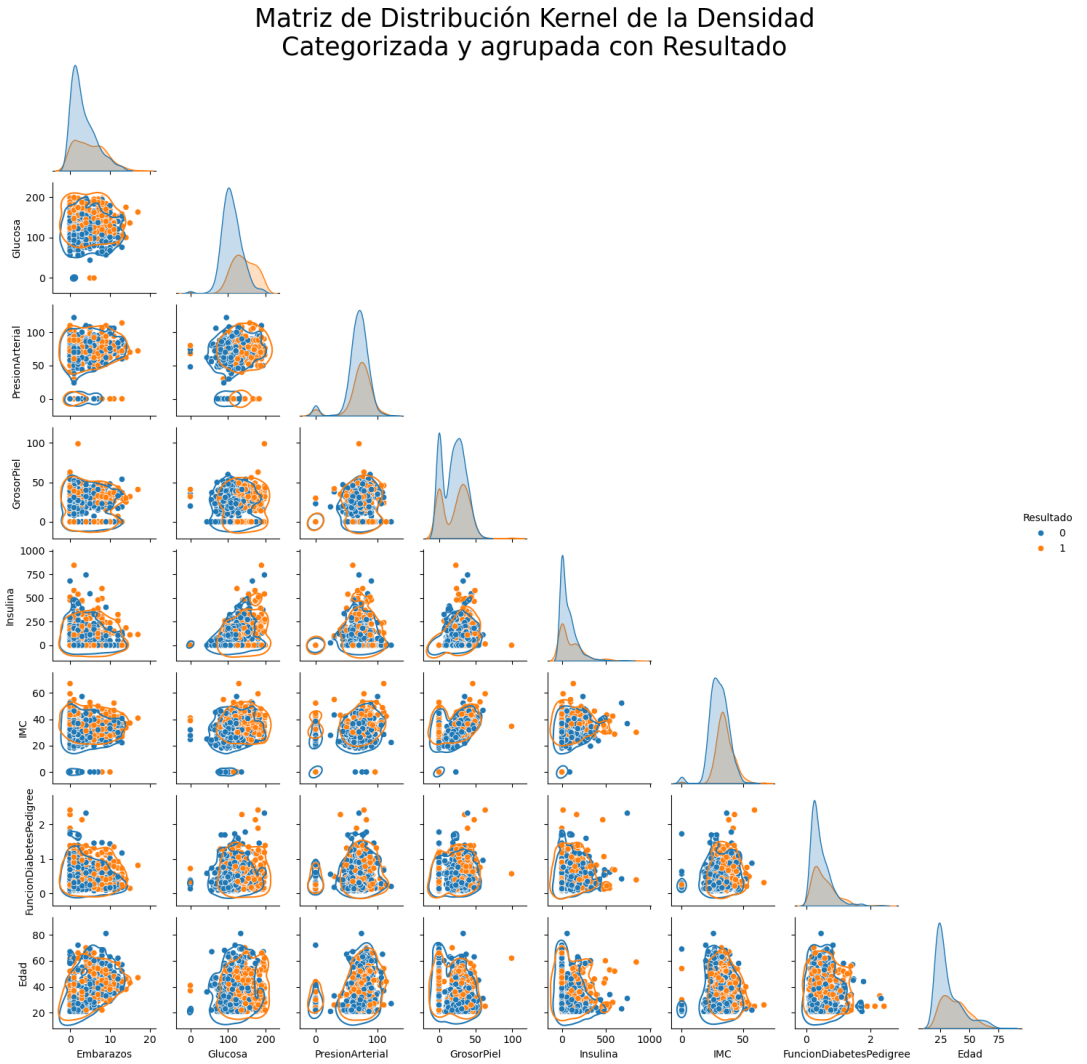

Figura 4.10: Matriz de Gráficos de Dispersión Dividido por Resultado y Clusterizando.

En la Figura 4.10, la mayoría de datos se encuentran dentro de ambas densidades para cada característica. Esto resulta beneficioso ya que se puede inferir que las instancias de 'Resultados' distan poco entre sí.

Preparación de los Datos

Se comienza por escalar los valores para, posteriormente, dividir en conjunto de entrenamiento y de prueba.

[14]:
```
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()
X = pd.DataFrame(sc_X.fit_transform(datos.drop(['Resultado'],axis = 1),),
columns=
['TotEmbarazos', 'Glucosa', 'PresiónArterial', 'GrosorPiel', 'Insulina',
'IMC', 'FuncionDiabetesPedigree', 'Edad'])
y = datos.Resultado
```

Como no se conoce un valor óptimo de vecinos se procede a utilizar la función `heuristico_knn()` con un máximo de 20 vecinos ($k = 20$) utilizando un método **heurístico**[2]

Implementación del Modelo

[13]:
```
k = 20
fun.heuristico_knn(k,X,y)
```

```
Puntuación máxima de entrenamiento 1.0 con k = [1]
Puntuación máxima de prueba 0.796875 con k = [7]
```

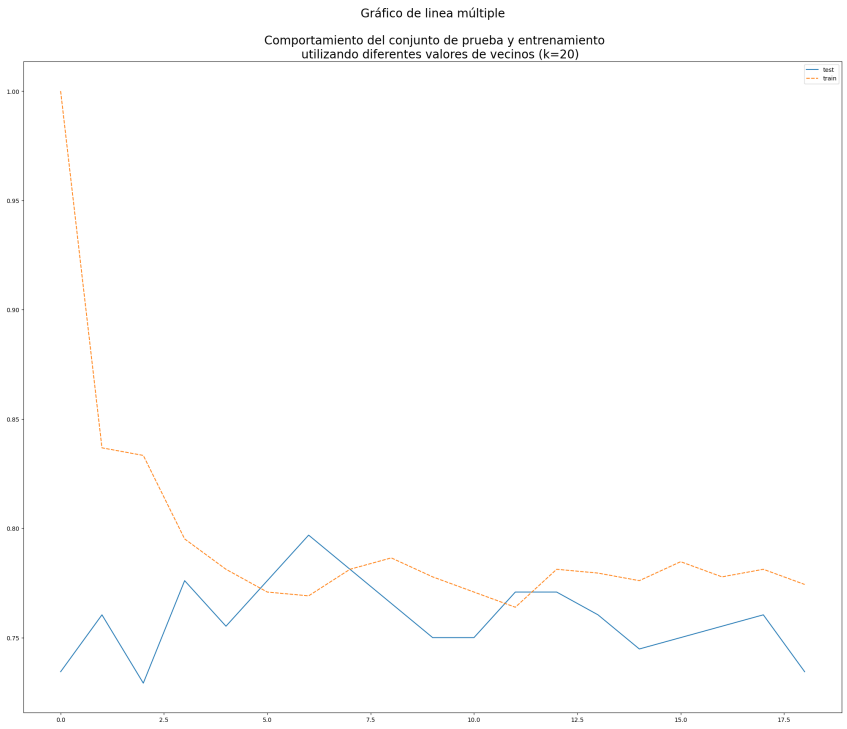

Figura 4.11: Método Heurístico.

Se nota en la Figura 4.11, que el valor de $k = 10$ es el mejor para la cantidad de vecinos que se necesitan para tener una precisión cercana a 0.8. Como se puede notar los valores de k y el puntaje entre ambos conjuntos distan mucho entre sí. Por lo que es conveniente visualizar estos puntajes en relación con los valores de k contrastando ambas gráficas entre sí, tal que:

Visualización de los Valores en Relación con su Puntaje

[2] El método heurístico contrapone ambos gráficos de línea para encontrar, tanto el punto donde se interceptan, como el comportamiento de los conjuntos.

[18]:
```
raw=pd.DataFrame([ks,test_p,train_p]).T
raw.columns=["ks","test_p","train_p"]
```

[18]:

Índice	ks	test_p	train_p
0	1	0.732824	1
1	2	0.725191	0.850575
2	3	0.740458	0.8659
3	4	0.748092	0.835249
4	5	0.763359	0.842912
5	6	0.763359	0.812261
6	7	0.793893	0.831418
7	8	0.78626	0.823755
8	9	0.770992	0.831418
9	10	0.78626	0.812261
10	11	0.793893	0.793103
11	12	0.824427	0.793103

Tabla 4.3: Tabla comparativa de puntajes con distinto número de vecinos.

Los datos de la Tabla 4.3 en la columna test_p representan el puntaje que se obtuvo aplicando el número de vecinos denotado por la columna ks con el conjunto de prueba.

Despliegue

Generando entonces el modelo de k-vecinos con $k = 7$ se obtienen los siguientes resultados:

Matriz de Confusión

[14]:
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(7)

fun.matrx_conf(X,y,model,1/4,0,(7,5))
```

```
Total de Aciertos: 146.
Total de Fracasos: 46.
Proporción de Aciertos: 76.04%.
```

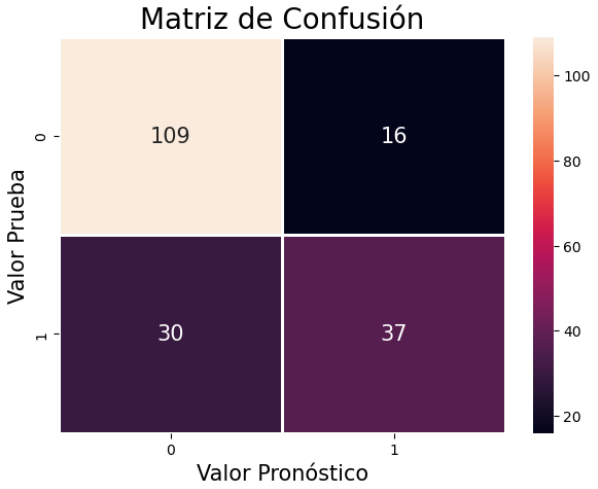

Figura 4.12: Matriz de Confusión.

Como se muestra en la Figura 4.12, se tienen un número significativo de valores calculados correctamente contra los que se obtuvo un resultado erróneo. Esto es 104 resultados correctos contra 27 errores, lo cual denota un 76 % de fiabilidad del modelo.

Curva de ROC

[15]:
```
fun.ROC(X,y,model,1/4,0)
```

```
Sin entrenar: ROC AUC=0.5.
Regresión Logística: ROC AUC=0.785.
```

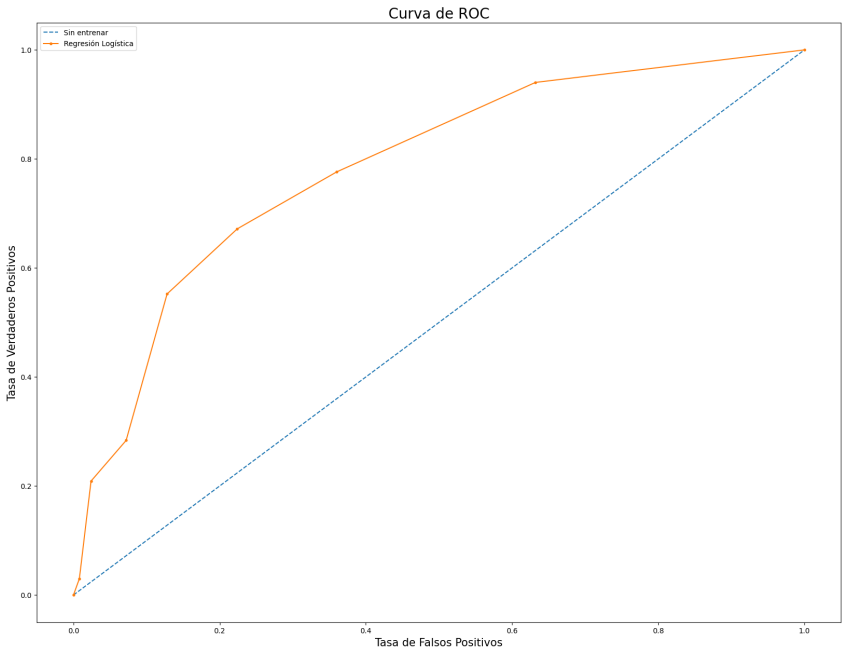

Figura 4.13: Curva de ROC.

Utilizando el espacio de ROC con $k = 7$ se puede notar que la *tasa de verdaderos positivos* crece antes de que la *tasa de falsos positivos* denote una sobrestimación del modelo. Además de que el área bajo la curva tiene un valor de 0.785; con lo cual se afirma que el modelo tiene un 78.5 % de confianza que acierte en el resultado.

Con esto, si se desea generar un modelo predictivo, se tiene que tomar en cuenta que es probable que la mayor cantidad de errores se encuentren en los falsos positivos; generando así un error tipo II. Pero, para este modelo en particular, el generar predicciones para conocer si el individuo pudiese adquirir diabetes; es mejor prevenir si es que se arroja un resultado de este tipo. Y, aunque exista la posibilidad de que el modelo se equivoque en ciertas ocasiones, en términos de salud la mejor decisión es pensar que el modelo no falla.

Conclusión

El modelo del k-vecino más cercano, para la búsqueda de patrones y generación de modelos de pronóstico, resulta muy útil cuando se tiene datos del tipo binario para analizar. Ya que, por definición del método, el uso de medidas (como la distancia) permite crear clusterizaciones óptimas para generar modelos fiables.

Aunque el uso de k-NN resulta factible para la búsqueda y reconocimiento de patrones, existen metodologías más acertadas para la implementación con datos del tipo categórico y/o binario.

Pero, para este ejemplo en particular, es notoria la forma de implementación con la metodología de k-vecino más cercano.

La vida es una sucesión de decisiones.

Kratos, God of War.

5

Árboles de Decisión

Los modelos basados en árboles de decisión suelen ser utilizados como un algoritmo supervisado no paramétrico; aunque existen también los *árboles de regresión*. En general los modelos tienen por meta la clasificación de instancias en función de variables predictivas particionando el espacio de características utilizando algo conocido como *reglas de división* [15]. El algoritmo secuencia preguntas del tipo si y sí no (*if/else if*) para la división de instancias y así generar la partición. Si es que el espacio de características no está diseñado para ser evaluado en forma dicotómica ($1, 0, Si, No, Verdadero, Falso,$ *etc.*), se utilizan criterios de orden ($a \leq b, b < a,$ *etc*) para la división de instancias.

5.1. Estructura

Un árbol de decisión tiene por partes [32]:

1. El **nodo raíz**. El cuál es de donde empieza la primera división del árbol y es el que contiene todos los datos de entrenamiento.
2. Los **nodos interiores**. Éstos son los nodos que se construyeron a partir del nodo raíz o de un nodo interior anterior.
3. **Ramas**. Son las conexiones entre los nodos.
4. Los **nodos hoja**. Generalmente son los resultados finales de crear un árbol de decisión.

Con esto se logra distinguir un número determinado de objetos que pertenecen al mismo grupo. Para ejemplificarlo mejor, en la Figura 5.1 se distingue entre tres tipos de pizzas (Vegetariana, Hawaiana y Nutella) primero se plantea una pregunta que divida las instancias del grupo. *¿Contiene carne?* divide en dos el conjunto de instancias ((Vegetariana, Nutella),(Hawaiana)). La siguiente división es con el nodo que contiene dos instancias. De modo que se plantea la pregunta de *¿Es postre?* a la instancia (Vegetariana, Nutella) transformándose en ((Vegetariana),(Nutella)) creando así el siguiente árbol:

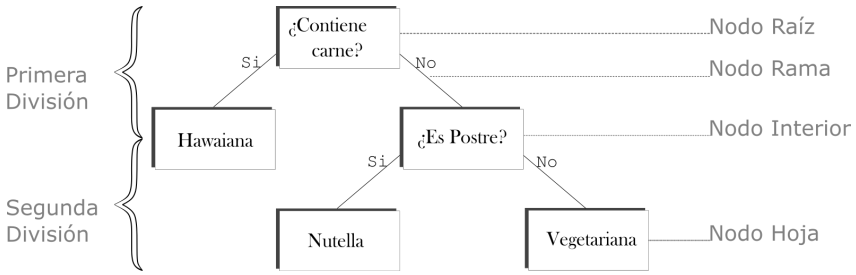

Figura 5.1: Ejemplo de lógica de agrupación en un árbol de decisión.

5.2. Método *CART*

El método conocido como *árbol de clasificación y regresión* (*classification and regression tree*, CART por sus siglas en inglés) divide el grupo de entrenamiento en subgrupos homogéneos y asigna una constante ajustada para cada subgrupo (o nodo). Los nodos se forman de manera recursiva utilizando particiones binarias sobre el espacio de características. Se realiza varias veces con el objetivo de encontrar la mejor característica $\hat{x}$ para dividir los datos entrantes en dos regiones R_1, R_2 de manera que el error general ε entre la respuesta real x y la constante de ajuste $c_k,\ k = 1, 2$ sea la mínima [5]. Con esto se identifican dos tipos de problemas:

1. Problema de regresión: los valores de respuesta promedio para todas las observaciones que caen en ese nodo.
2. Problema de clasificación: la clase que tiene representación mayoritaria.

Para problemas de regresión se utiliza la *suma de cuadrados residuales* (SCR) como se muestra a continuación:

$$\text{SCR} = \sum_{i \in R_1} (y_i - c_1)^2 + \sum_{i \in R_2} (y_i - c_2)^2 . \tag{5.1}$$

Para problemas de clasificación la partición generalmente se hace para maximizar la reducción con *entropía cruzada* o el *índice de Gini* [27] [35].

La *entropía cruzada* es una métrica donde al error cuadrático medio se le agrega el logarítmo de la probabilidad haciendo producto con la clase de valor verdadero. Ésto hace que la entropía cruzada sea especialmente buena para no caer en el error de una respuesta incorrecta. La ecuación para el cálculo de la entropía está dada por:

$$\text{Entropía} = \mathbb{E} = -p \log_2 (p) - q \log_2 (q)$$

donde $p = 1 - q$ representa la probabilidad de éxito.

El *índice de Gini*, también conocido como *medida de pureza*, muestra donde un valor pequeño indica que un nodo contiene de manera predominante observaciones de una sola clase, denominada etiqueta. La ecuación para el cálculo del índice de Gini está dado por:

$$\text{Gini} = \mathbb{G} = 1 - \sum_i (p_i)^2 \tag{5.2}$$

donde p_i representa la probabilidad de cada etiqueta con índice i.

5.3. Profundidad

Se le conoce como profundidad a la cantidad niveles en las que se van a generar los nodos hoja. Mientras más complejas sean las ramas del árbol es más probable llegar a un sobre-ajuste del

modelo. Por esto mismo se tiene que encontrar la mejor profundidad para el árbol; para ello es común utilizar metodologías de parada como la *detención anticipada* y el método de podar (conocido como *pruning* en inglés) [5] [27].

Detención Anticipada

La detención anticipada restringe de manera explícita el crecimiento del árbol, dos de los enfoques más comunes son el restringir la profundidad hasta cierto nivel o restringir el número mínimo de observaciones dentro de los nodos. En lo que respecta la restricción a la profundidad del árbol se toma en cuenta que, mientras más superficial es el árbol, menos variaciones tendrá el modelo para predecir; sin embargo, puede existir que en algún momento, al no tener suficiente profundidad en el árbol, exista mucho sesgo ya que no se pueden capturar patrones complejos en los árboles muy superficiales. De otro modo, al restringir el tamaño mínimo de las observaciones dentro de los nodos se opta por no dividir los nodos intermedios que tienen muy pocos datos en su interior.

Podado

Otra alternativa es la técnica o metodología de *podado*, la cual se basa en crear un árbol grande y complejo para después *podar* las ramas y los nodos interiores para encontrar un sub-árbol óptimo. Para encontrar dicho sub-árbol se necesita un parámetro que se denomina de *complejidad* denotado por α la cual penaliza a la SCR para un número T de nodos terminales. De modo que lo que se busca es minimizar la SCR sumando al producto de los nodos T con el parámetro de complejidad α de la siguiente manera:

$$\text{minimizar}\left\{SCR + \alpha|T|\right\}. \tag{5.3}$$

Esto es, dado un valor de α, se encuentra el árbol podado más pequeño con el error de penalizado SCR más bajo. y para hallar el mejor α se suelen utilizar modelos cambio de variable para encontrar el valor óptimo del mejor sub-árbol proveniente de la poda. A medida que un árbol crece la reducción del SCR debe ser mayor que la penalización por complejidad.

NOTA: Dentro de la práctica, la manera más utilizada para saber que tan fiable es nuestro modelo de árbol de decisión es conocida como la *matríz de confusión* la cual es una matriz de orden 2 donde la suma de la diagonal principal proporciona el número de aciertos dentro de la fase de testeo del modelo. Esto resulta muy útil debido a que, si dicha matriz no cumple un porcentaje de aciertos, se opta por modificar el árbol [5].

Algoritmo

1. Crear el **nodo raíz** R_y y asociar todas las características (o instancias) a dicho nodo. Se denota a $X := R_y$ como el conjunto de nodos que se van a dividir.
2. Si $X = \emptyset$, se devuelve el árbol con raíz r y se finaliza el método.
3. Seleccionar una característica $x \in X$ y sacarlo del conjunto para determinar la puntuación del nodo con la función divisora que se use.
4. Determinar si dicha división es posible/necesaria; de lo contrario regresar a 2.
5. Para todos los atributos divisores posibles, evaluar los efectos de la división en dichos atributos y seleccionar el atributo que proporcione el mejor ajuste.
6. Si la mejora es suficiente y necesaria, crear un conjunto de nodos secundarios X', agregar X' a X y conectar la característica x a todos los nodos secundarios en X'
7. Asociar cada nodo en X' a su conjunto de instancias correspondiente e ir al paso 2 [36].

5.4. Ejemplo con Python

Árboles de Decisión

El grupo de características es el mismo utilizado en el capítulo anterior. Al ser ambos métodos de clasificación, son notorias las diferencias al aplicar diferentes metodologías.

Bibliotecas

[1]:
```
import pandas as pd # Paquetería para cargar archivos
import numpy as np # Paquetería de computo científico
import Catalogo_de_Funciones as fun # Paquetería del catálogo de funciones
from sklearn.preprocessing import StandardScaler # Escalar valores
from sklearn.tree import DecisionTreeClassifier # Paquetería para árboles
```

[2]:
```
path = 'diabetes.csv'
fun.head_describe_csv(path)
```

Característica	count	mean	std	min
Embarazos	768	3.845052	3.369578	0
Glucosa	768	120.894531	31.972618	0
PresiónArterial	768	69.105469	19.355807	0
GrosorPiel	768	20.536458	15.952218	0
Insulina	768	79.799479	115.244002	0
IMC	768	31.992578	7.88416	0
FuncionDiabetesPedigree	768	0.471876	0.331329	0.078
Edad	768	33.240885	11.760232	21
Resultado	768	0.348958	0.476951	0

Característica	25 %	50 %	75 %	max
Embarazos	1	3	6	17
Glucosa	99	117	140.25	199
PresiónArterial	62	72	80	122
GrosorPiel	0	23	32	99
Insulina	0	30.5	127.25	846
IMC	27.3	32	36.6	67.1
FuncionDiabetesPedigree	0.24375	0.3725	0.62625	2.42
Edad	24	29	41	81
Resultado	0	0	1	1

Tabla 5.1: Tabla con estadísticos generales.

En la Tabla 5.1 existen valores sin una interpretación clara; como 0 en el mínimo de glucosa. Identificadas las características donde se presentan este tipo de valores se procede a reemplazarlos por un valor nulo para facilitar la depuración:

La Tabla 5.2 muestra la cantidad de valores nulos hallados en la base de datos por cada característica. Dichos valores son eliminados de la base de datos para continuar con la implementación. Así, con esta nueva base de datos, se continua por explorar de manera gráfica. Se utilizan diagramas de violín, los cuales son una forma más descriptiva de los datos que el diagrama de cajas y bigotes.

Exploración de Datos

NoEmbarazos	0
Glucosa	5
PresiónArterial	35
GrosorPiel	227
Insulina	374
IMC	11
FDP	0
Edad	0
Resultado	0

Tabla 5.2: Frecuencia de valores nulos en la base de datos.

Matriz de Correlación

[4]:
```
data = data.reset_index() # re indexado
data = data.drop("index", axis=1)

fun.MatrizCorrelación(data)
```

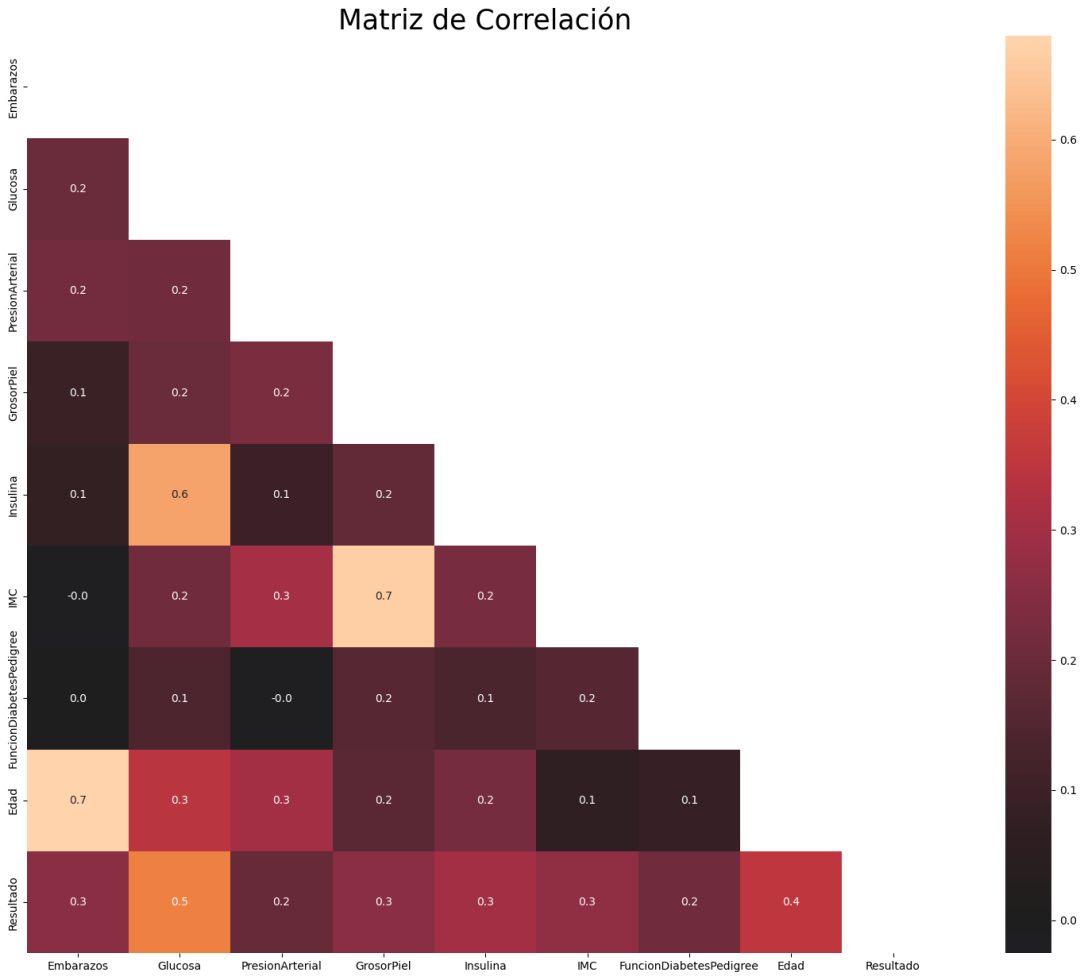

Figura 5.2: Matriz de correlación.

[5]:
```
X = data.iloc[:,:-1]
y = data.iloc[:,-1:]
```

[6]:
```
fun.MatrizViolines(4,2,X,y,data)
```

Es de notar que no se grafican los valores de la columna de resultados ya que son dicótomos.

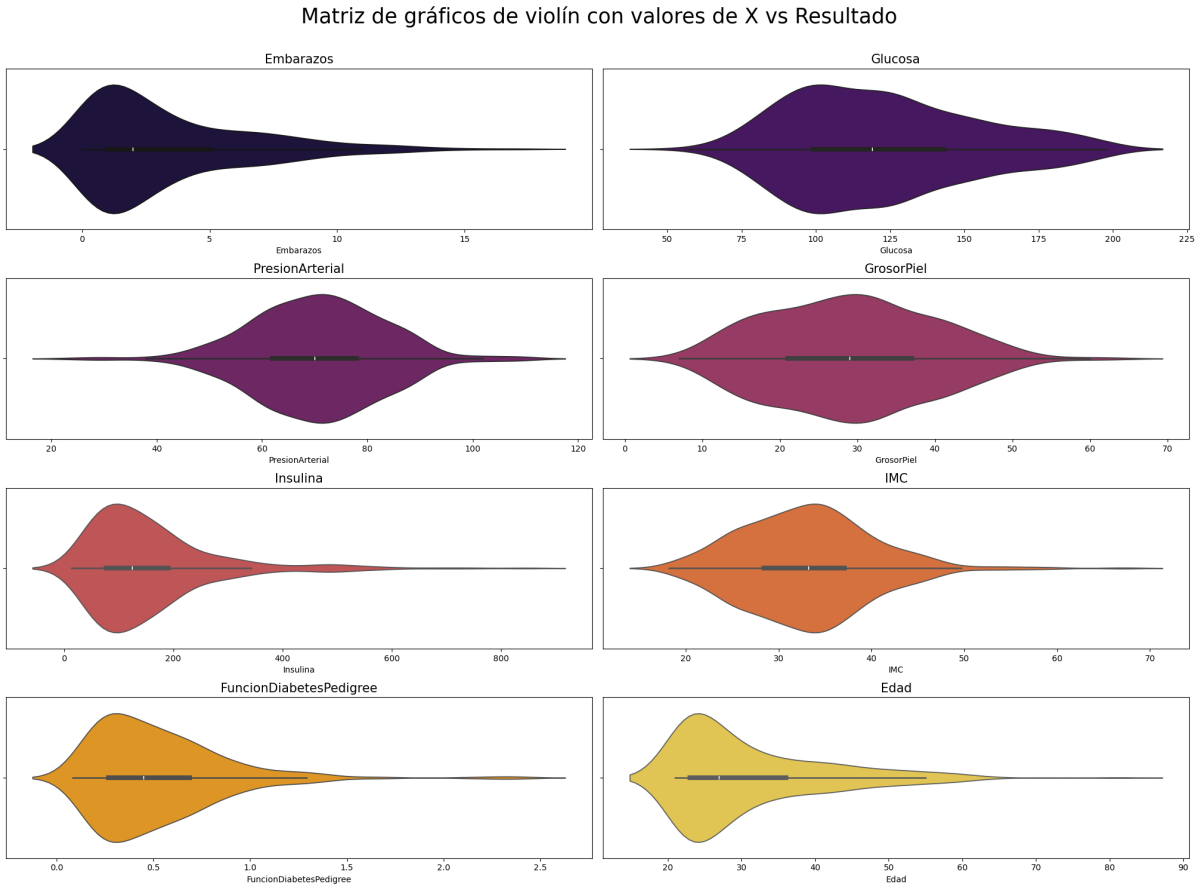

Figura 5.3: Matriz de Diagramas de Violín Mostrando la Distribución de los Valores de las Características.

La Figura 5.3 muestra una matriz de gráficos de violín con todas las características de la base de datos. Cada uno de los gráficos se construye con una gráfica de cajas y bigotes en el interior y la distribución de los valores a cada extremo. Con esto se analizan de mejor manera los datos. Por ejemplo, el gráficos de insulina y función diabetes pedigree, a primera vista, presentan una cantidad notable de datos atípicos acumulados entre los valores de $[400, 600]$ y de $[1.25, 2.5]$ respectivamente. Aunque a priori se consideren datos atípicos, el eliminar a los individuos dentro de ese intervalo afecta de manera directa a las demás características. Para eliminar individuos de manera correcta, y no caer en el error tipo 1, se requiere un análisis más a fondo del comportamiento de los valores y los individuos entre sí.

Además, se pueden analizar características de manera separada y especificando cada vez lo que se quiere visualizar.

[7]:
```
X_col = 'Resultado'
y_col = 'Insulina'

fun.Violin(data, X_col, y_col, (10,5))
```

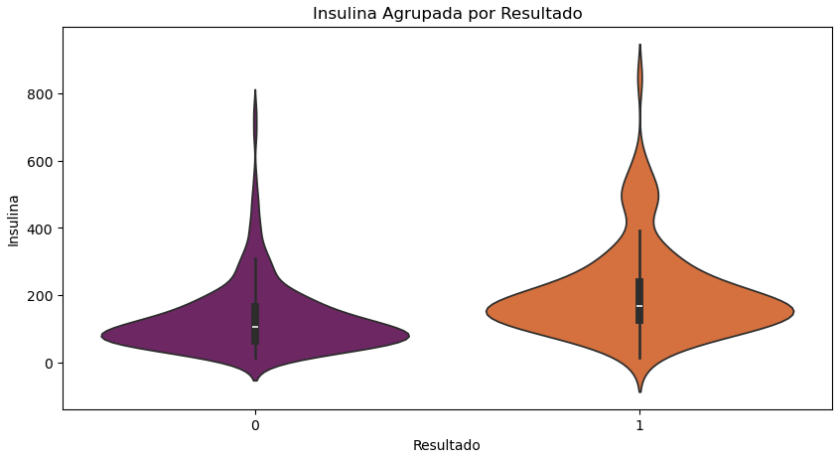

Figura 5.4: Diagramas de Violín que muestra la Distribución de la Insulina basándose en el Resultado.

La Figura 5.4 muestra dos diagramas de violín, tomando a los individuos con puntuación 0 en la característica de resultado en el diagrama de la izquierda (naranja) y a los individuos con puntuación 1 en el diagrama de la derecha (guinda). De este modo se pueden observar los cuartiles y la distribución kernel estimada de los valores de insulina divididos en los dos grupos.

Esto es de utilidad a la hora de contrastar características entre sí. Por ejemplo, se puede observar en los individuos con puntaje cero una mediana más cercana alrededor de 100, que en el caso contrario los de puntaje uno la mediana está más cercana al valor de 200.

Si se quisiese partir ambos diagramas de violín se utiliza el argumento "hue", igual con una característica dicotómica como argumento. La creación de valores dicotómicos siempre es a criterio de quien realiza el análisis. En este caso sólo se utilizarán las columnas con las etiquetas de glucosa, presión arterial e insulina para crear tres nuevas columnas donde se denota si el individuo en cuestión se encuentra por encima de la media de cada columna mencionada respectivamente. Es decir, a todos los individuos se le asignan tres valores booleanos nuevos, correspondientes a las tres columnas nuevas: glucosa arriba de la media, presión arterial arriba de la media y nivel de insulina por encima de la media:

[8]:
```
v1 = []
for u in range(0, len(data.Glucosa)):
        if data['Glucosa'].values[u] >= data['Glucosa'].values.mean():
                v1.append(1)
        elif data['Glucosa'].values[u] < data['Glucosa'].values.mean():
                v1.append(0)

data = data.assign(Clasificacion_Media_Glucosa = v1)

v1 = []
for u in range(0, len(data['PresionArterial'])):
        if data['PresionArterial'].values[u]>=
```

```
        data['PresionArterial'].values.mean():
                v1.append(1)
        elif data['PresionArterial'].values[u]<
        data['PresionArterial'].values.mean():
                v1.append(0)

data = data.assign(Clasificacion_Media_PresionArterial = v1)

v1 = []
for u in range(0, len(data.Insulina)):
        if data['Insulina'].values[u] >= data['Insulina'].values.mean():
                v1.append(1)
        elif data['Insulina'].values[u] < data['Insulina'].values.mean():
                v1.append(0)

data = data.assign(Clasificacion_Media_Insulina = v1)
```

Teniendo ya las nuevas columnas se procede con más diagramas de violín para continuar con la exploración. Los diagramas suelen colocarse dentro de métricas coherentes teniendo en cuenta una característica que las divida en dos subconjuntos disjuntos del grupo de característica. Si se desea conocer, en este caso, como se comporta la insulina dentro de toda la muestra, los valores de la característica de resultado es perfecto para realizar esta partición:

[9]:
```
h = "Clasificacion_Media_Glucosa"

fun.ViolinHue(data,X_col,y_col,h,(10,5))
```

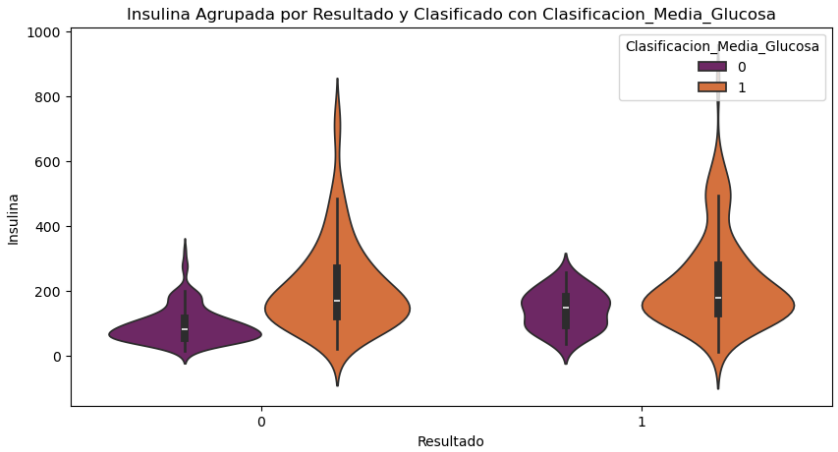

Figura 5.5: Diagramas de Violín que muestra la Distribución de la Insulina basándose en el Resultado Particionado por la Media de la Glucosa.

En la Figura 5.5, tomando como ejemplo los violines azules, es notorio el desplazamiento de la mediana hacia el valor de 200 y en el otro par de diagramas (verdes) se desplaza hacia 100; así

entonces se plantean ciertas dudas. Por ejemplo si es que *existe una relación directa entre los niveles de insulina y la cantidad de glucosa del individuo.*

Aunque es una herramienta poderosa los diagramas de violín, resulta un tanto redundante y confuso tener cuatro gráficos que expliquen el comportamiento de una característica sujeta a dos particiones (o splits), de modo que se suele utilizar el argumento *split* = *True* para que los violines se indexen en términos del argumento de "hue". Dando como resultado dos diagramas de violín:

[10]:
```
v = True

fun.ViolinHueSplit(data,X_col,y_col,h,v)
```

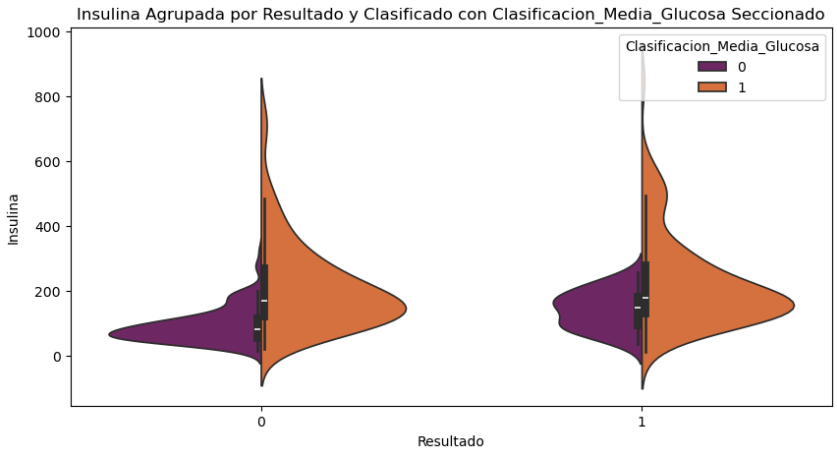

Figura 5.6: Diagramas de Violín que muestra la Distribución de la Insulina basándose en el Resultado Particionado por la Media de la Glucosa Mostrado en Dos Gráficos.

Al trabajar los datos con diagramas de violín se pueden notar características que se relacionan. Como con el gráfico inicial de la Figura 5.4 que guarda mucha relación con el diagrama de la Figura 5.5.

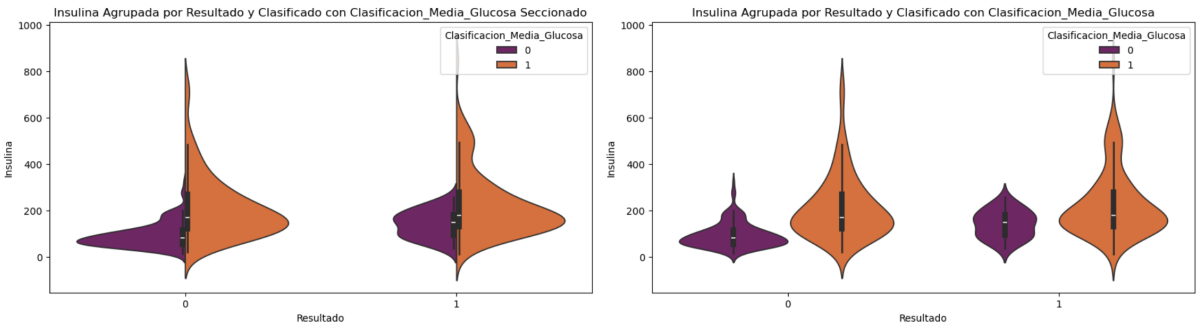

Figura 5.7: Comparación de Gráficos de Violines.

En la Figura 5.7 se comparan dos gráficas de violín, la de la Figura 5.5 y la Figura 5.6. Es de notar que la medianas de ambas gráficas se corresponden, la diferencia radica en la segmentación de la gráfica de la derecha que muestra la distribución de individuos por encima de la media de la glucosa, comparados con el valor de los resultados.

De este modo se observa con mayor claridad como afectan los individuos entre sí en diferentes características. De modo que, si en un principio se hubiera optado por eliminar a los individuos con insulina superior a 400 igual se eliminarían una cantidad significativa de individuos que tienen la glucosa por encima de la media.

[11]:
```
h = "Clasificacion_Media_PresionArterial"

fun.MatrixViolinesHueSplit(data,X,y,h,v=True,rows=4,cols=2,size = (20,15))
```

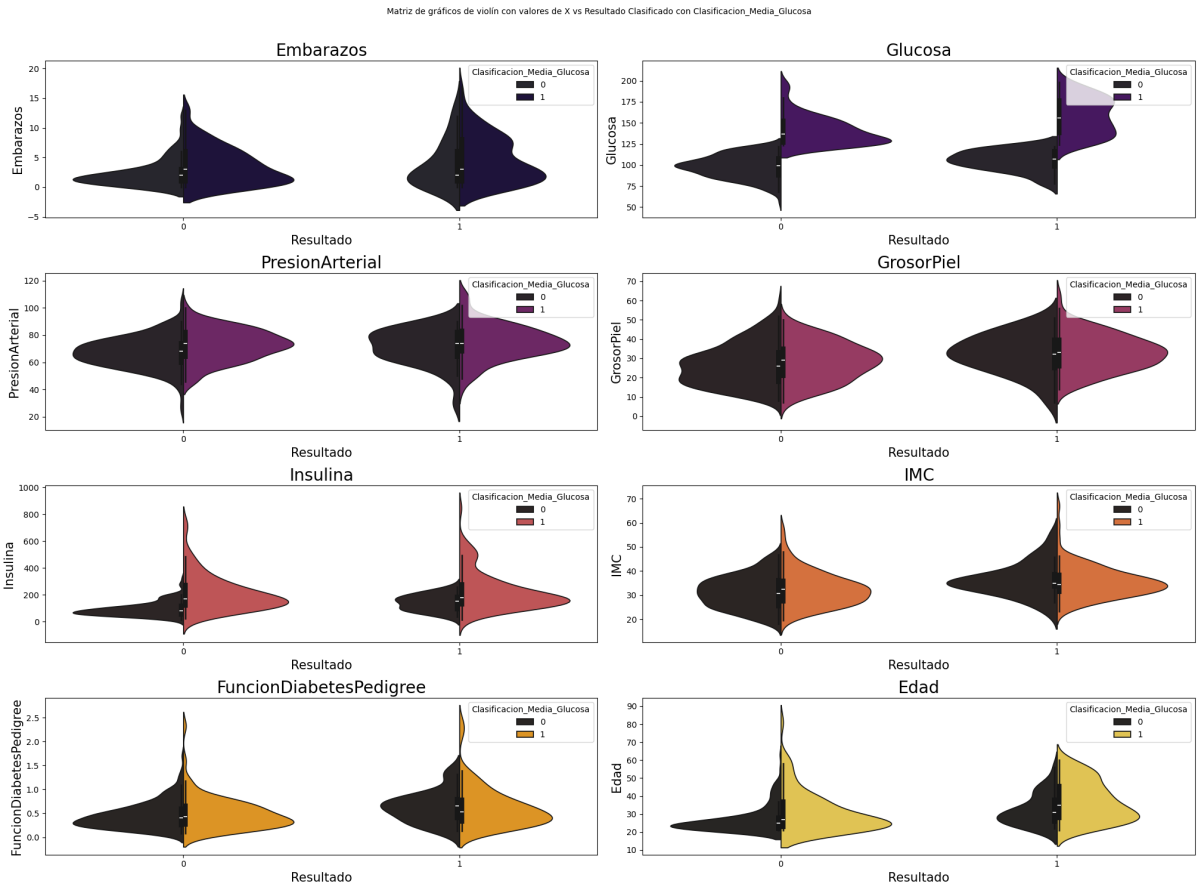

Figura 5.8: Matriz de diagramas de violín segmentadas mediante la media de la insulina agrupadas por el resultado.

En la Figura 5.8 la matriz de gráficos contiene, en cada entrada, la segmentación de los valores de las características mediante la media de la glucosa y clasificados en individuos con resultado 0 o 1. De esta manera se localizan características que pudieran estar correlacionadas de manera directa. Por ejemplo, el diagrama de posición $(2, 2)$ (fila 2, columna 2) muestra dos diagramas de violín interesantes:

- El primero, de izquierda a derecha, representa a las personas a las cuales no se les detectó diabetes. Esta gráfica muestra una distribución bimodal para las personas que se encuentran por debajo de la media de la glucosa y una distribución unimodal para quienes tiene la glucosa por encima de la media. De igual manera se nota que alrededor de la mediana, el grosor de piel de las personas que no viven con diabetes es similar a las que si lo presentan; es decir, ambas se observan con poca dispersión.

- El segundo, que representa a las personas a las cuales se les detectó diabetes, muestra distribuciones similares por encima y por debajo de la media de la glucosa. Se puede inferir que, conociendo el grosor de la piel, se conoce un aproximado del nivel de glucosa del individuo.

De esta manera se pueden plantear, observando la parte descriptiva, hipótesis como la anterior.

[13]:
```
h = "Clasificacion_Media_PresionArterial"

fun.MatrixViolinesHueSplit(data,X,y,h,v=True,rows=4,cols=2,size = (20,15))
```

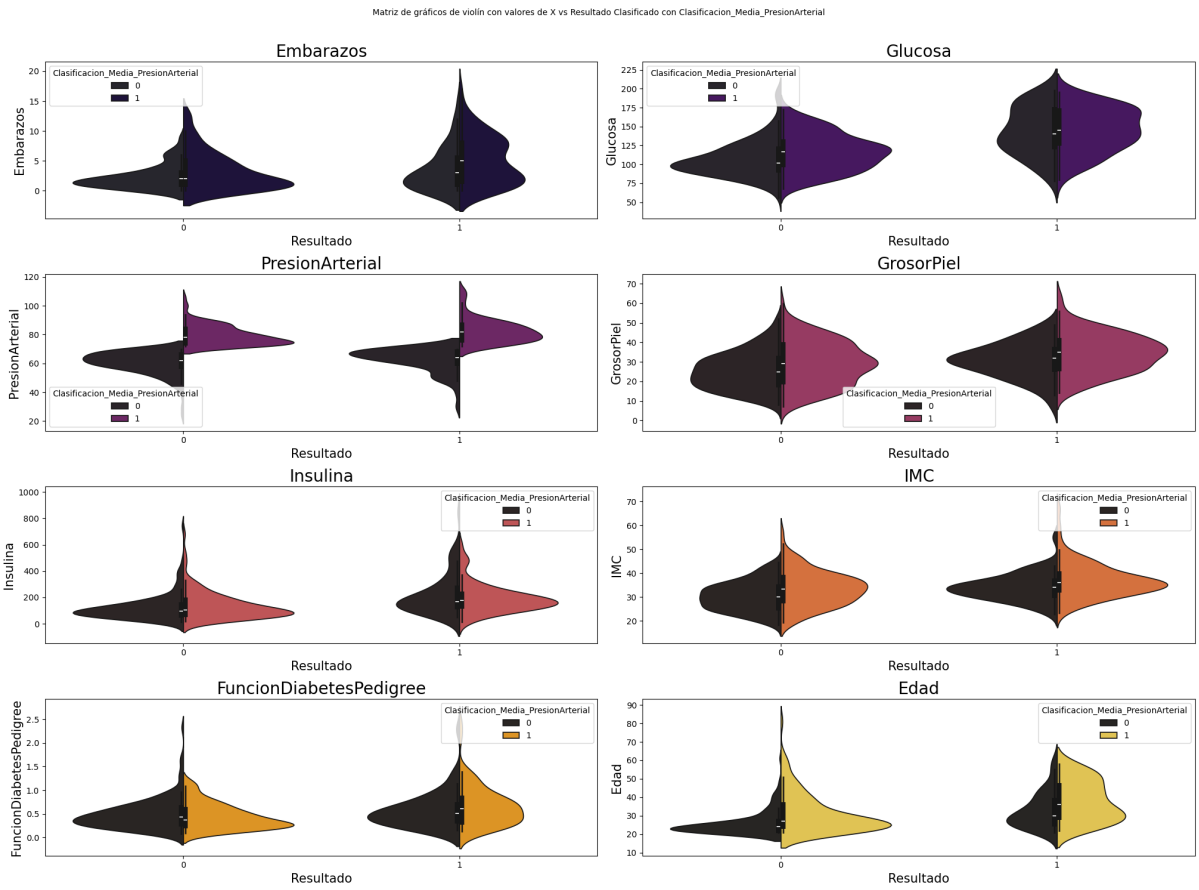

Figura 5.9: Matriz de diagramas de violín segmentadas mediante la media de la insulina, glucosa y presión arterial agrupadas por el resultado.

En la Figura 5.9 el diagrama de posición $(3, 1)$ (fila 3, columna 1) muestra dos diagramas de violín interesantes:

- El primero de izquierda a derecha, que representa a las personas a las cuales no se les detectó diabetes, muestra que las distribuciones son similares. Con lo que se puede inferir que, alrededor de la mediana, los niveles de insulina son más similares para personas que se encuentren por encima o por debajo de la media de la presión arterial.

- El segundo de izquierda a derecha, que representa a las personas a las cuales se les detectó diabetes, muestra una distribución leptocúrtica para los individuos que se encuentran con una presión arterial por encima de la media. La mayoría de individuos están alrededor de la media y de la mediana de insulina

[14]:
```
h = "Clasificacion_Media_Glucosa"

fun.MatrixViolinesHueSplit(data,X,y,h,v=True,rows=4,cols=2,size = (20,15))
```

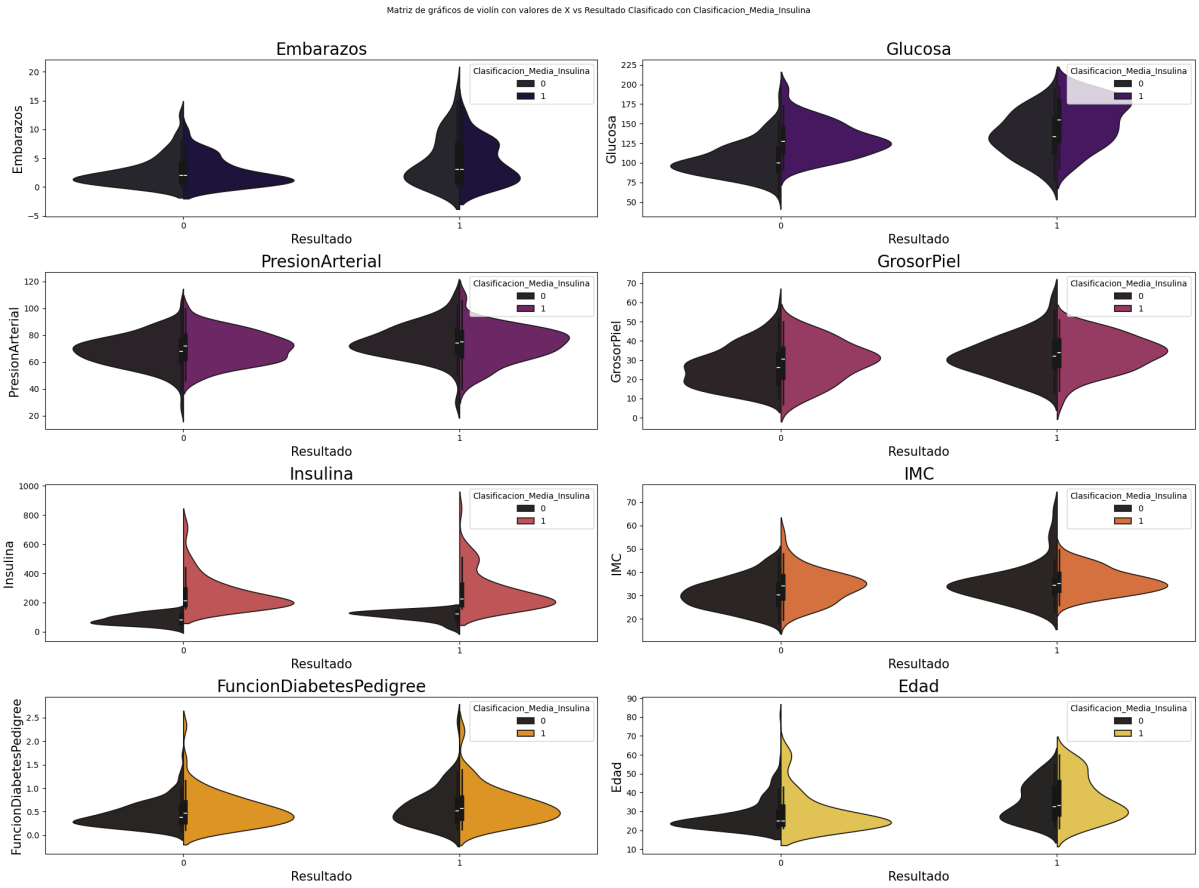

Figura 5.10: Matriz de diagramas de violín segmentadas mediante la media de la insulina, glucosa y presión arterial agrupadas por el resultado.

En la Figura 5.10 el diagrama de posición $(2, 1)$ (fila 2, columna 1) muestra dos diagramas de violín interesantes:

- El primero de izquierda a derecha, que representa a las personas a las cuales no se les detectó diabetes, muestra que los individuos con valores de insulina por debajo de la media tienen un valor medio de glucosa alrededor del 100. Mientras que, los individuos que están por encima de la media de la insulina, tienen una media de glucosa alrededor de 130.

- El segundo de izquierda a derecha, que representa a las personas a las cuales se les detectó diabetes, las personas que están por debajo de la media de la insulina, la distribución de

la glucosa se extiende más sobre los valores del eje vertical. Mientras que, los individuos que están por encima de la media de la insulina, muestra una distribución bimodal cerca de los valores más altos de la glucosa.

Implementación del Modelo

[35]:
```
SS_ = StandardScaler()

df = data.iloc[:,0:9]

X=pd.DataFrame(SS_.fit_transform(df.drop(['Resultado'],axis=1),),columns=
['Embarazos','Glucosa','PresionArterial',
'GrosorPiel','Insulina','IMC','FuncionDiabetesPedigree','Edad'])

y=data.Resultado
```

Dentro del entrenamiento del modelo se busca utilizar el mejor método para generar el árbol. Para ello se estudia el comportamiento de los conjuntos de prueba y entrenamiento utilizando el índice de Gini y la Entropía como sigue:

índice de Gini

[37]:
```
testsize = 1/4
criterion = 'gini'
depth = 3
rs = 0

etiquetas = data.iloc[:,:-4].columns

fun.Arbol(X,y,etiquetas,testsize,criterion,depth,rs)
```

```
Árbol de decisión
Criterio: gini.
Profundidad: 3.
Estadísticos:
Precisión:0.8163
Puntuación Conjunto de Entrenamiento:0.8095
Puntuación Conjunto de Prueba: 0.8163
Diferencia Absoluta de la Puntuación de los Conjuntos:0.0068
```

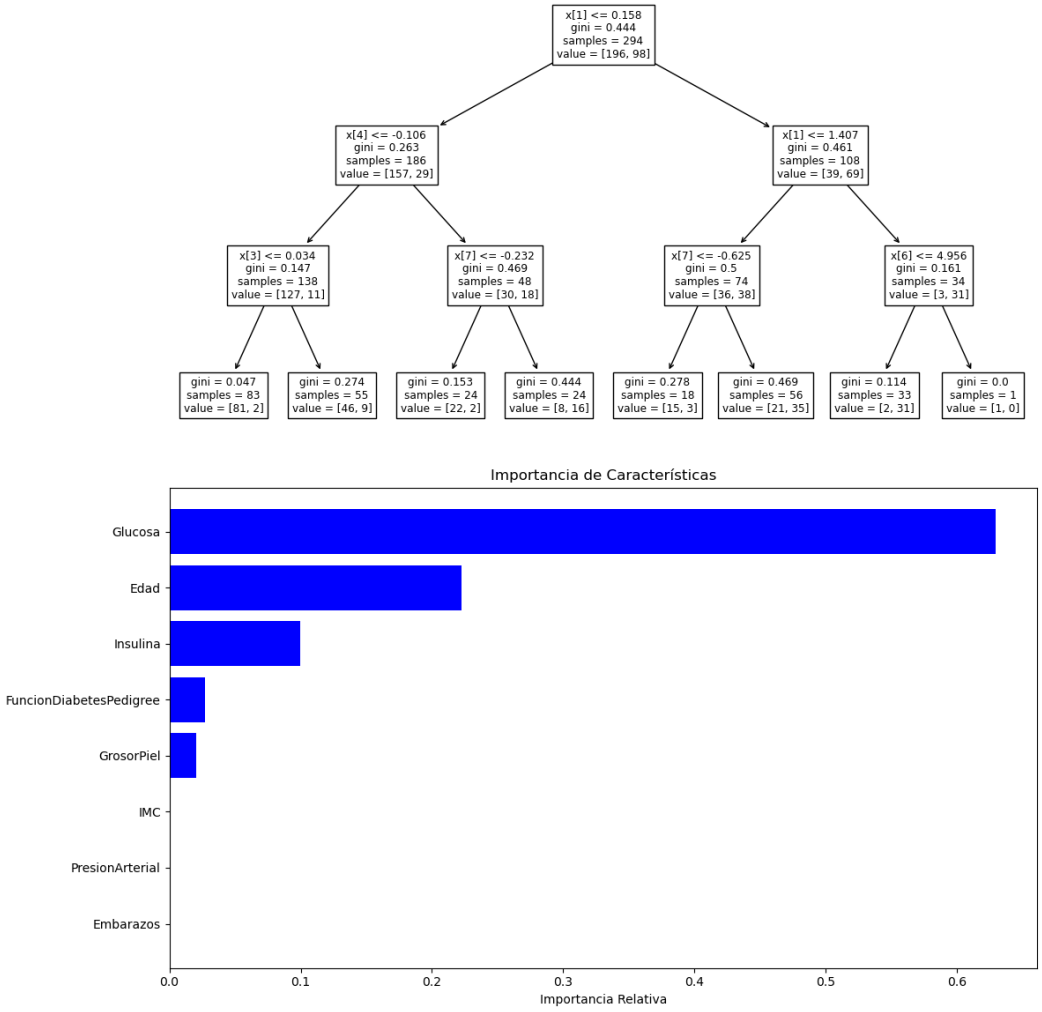

Figura 5.11: Árbol de decisión e importancia relativa generado mediante el índice de Gini.

Matriz de confusión

[38]:
```
CriterioClasif = DecisionTreeClassifier(
criterion = criterion, max_depth = depth, random_state = rs)

fun.matrx_conf(X,y,CriterioClasif,testsize,rs)
```

```
Total de Aciertos: 75
Total de Fracasos: 23
Proporción de Aciertos: 76.53%
```

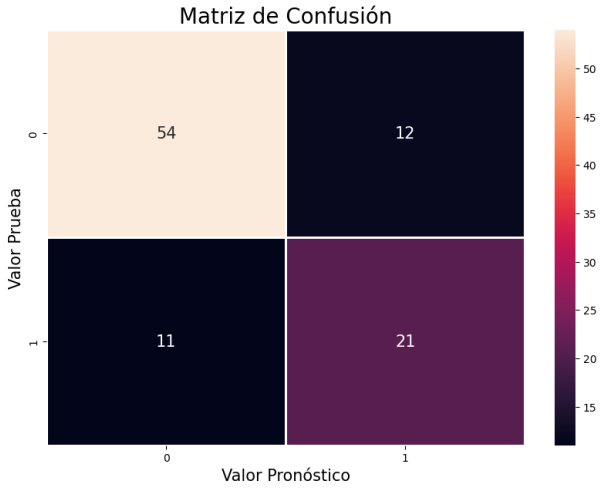

Figura 5.12: Matriz de confusión asociada al modelo de árbol con índice de Gini.

La Figura 5.12 muestra una matriz de confusión que se resume en la siguiente tabla:

	0	1	Totales
Aciertos	54	12	66
Fracasos	11	21	33

Tabla 5.3: Tabla resumen de la matriz de confusión.

Curva de ROC

[39]:
```
fun.ROC(X,y,CriterioClasif,testsize,0)
```

```
Sin entrenar: ROC AUC=0.500
Regresión Logística: ROC AUC=0.707
```

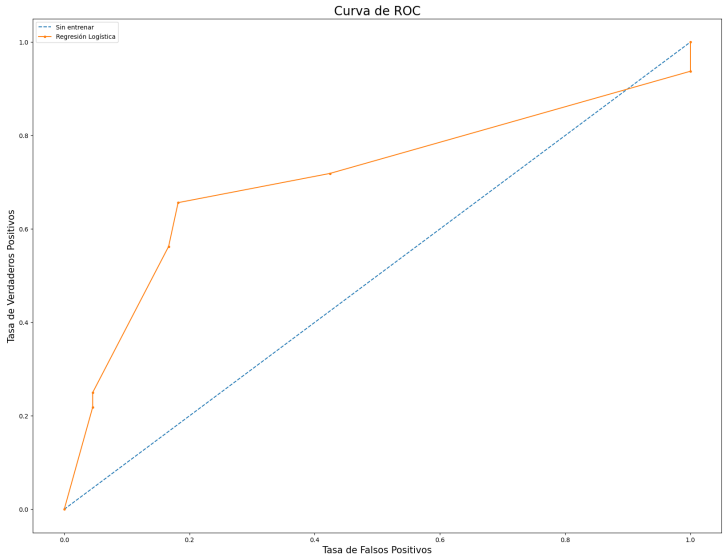

Figura 5.13: Curva de ROC

La Figura 5.13 muestra la curva de ROC generada con el índice de Gini. Es de notar que, aunque

la curva en un inicio crece, casi al final de la curva se aprecia un descenso. Esto implica una aproximación deficiente generando más falsos positivos en la implementación.

Entropía

[40]:
```
testsize = 1/4
criterion = 'entropy'
depth = 3
rs = 0
etiquetas = data.iloc[:,:-4].columns
MC_Ent =['Glucosa','Edad',
'FuncionDiabetesPedigree','IMC'] #Mejores Características

fun.Arbol(X,y,etiquetas,testsize,criterion,depth,rs)
```

```
Árbol de decisión
Criterio: entropy.
Profundidad: 3
Estadísticos:
Precisión:0.7449
Puntuación Conjunto de Entrenamiento:0.8129
Puntuación Conjunto de Prueba: 0.7449
Diferencia Absoluta de la Puntuación de los Conjuntos:0.0680
```

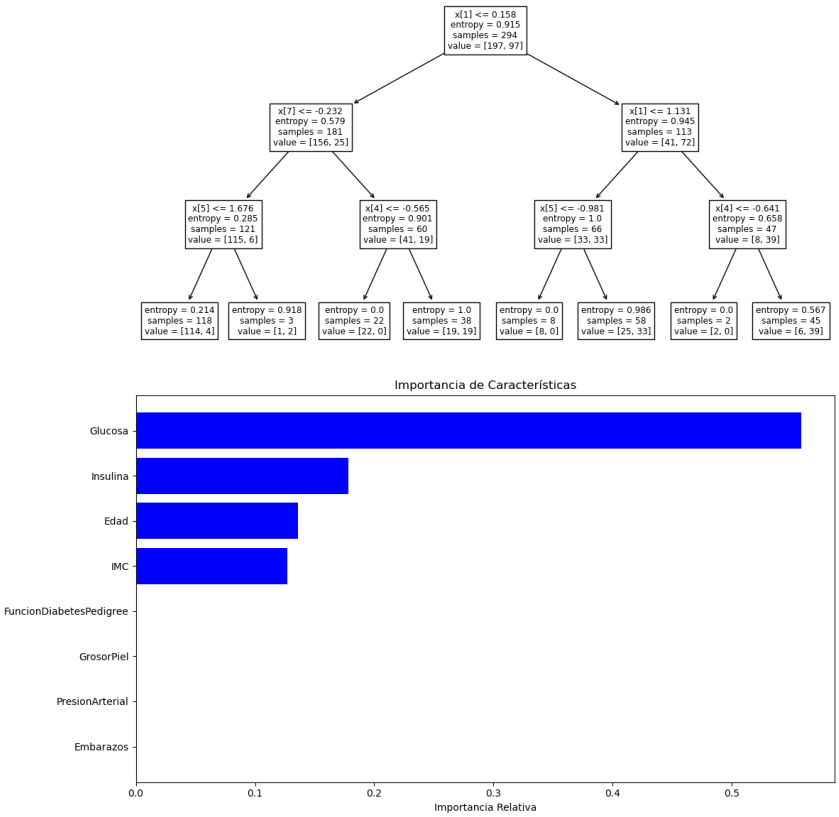

Figura 5.14: Árbol de decisión e importancia relativa generado mediante la entropía.

Matriz de confusión

[41]:
```
CriterioClasif = DecisionTreeClassifier(
criterion = criterion,max_depth=depth,random_state=rs)

fun.matrx_conf(X,y,CriterioClasif,testsize,rs)
```

```
Total de Aciertos: 76
Total de Fracasos: 22
Proporción de Aciertos: 77.55%
```

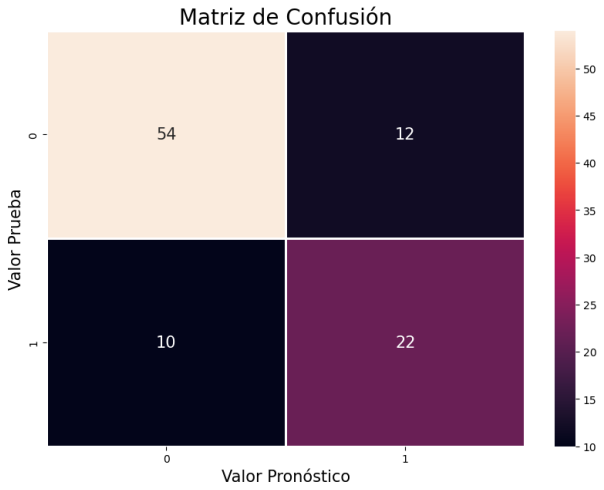

Figura 5.15: Matriz de confusión asociada al modelo de árbol con Entropía.

La Figura 5.15 muestra una matriz de confusión que se resume en la siguiente tabla:

	0	1	Totales
Aciertos	54	12	66
Fracasos	10	22	32

Tabla 5.4: Tabla resumen de la matriz de confusión.

Curva de ROC

[42]:
```
fun.ROC(X,y,CriterioClasif,testsize,0)
```

```
Sin entrenar: ROC AUC=0.500
Regresión Logística: ROC AUC=0.782
```

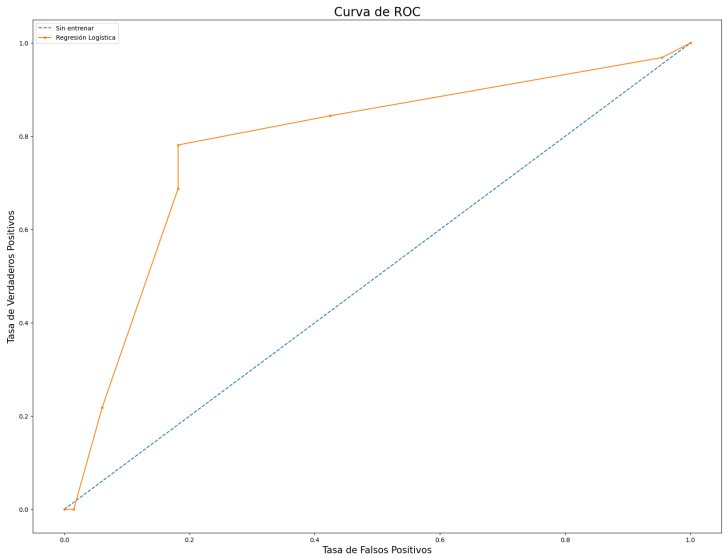

Figura 5.16: Curva de ROC.

La Figura 5.16 muestra la curva de ROC con la implementación con entropía. Es de notar que, al principio de la curva se muestra un descenso hacía la tasa de falsos positivos; pero, esto se compensa teniendo una curva más consistente en la implementación del modelo.

Esto podría indicar que, la implementación con la entropía, es mejor para generar el modelo predictivo.

Ahora, dentro de las Figuras 5.11 y 5.14 existe una gráfica con la importancia relativa de cada característica para la implementación del modelo. Esto indica que, los valores más importantes para la implementación son únicamente:

- índice de Gini:
  - Glucosa.
  - Edad.
  - Insulina.
  - Función Diabetes Pedigree.
  - Grosor Piel
- Entropía:
  - Glucosa.
  - IMC.
  - Edad.
  - Insulina.

Es común que, a la hora de comparar las puntuaciones obtenidas en las particiones experimentales, estas discrepen bastante a la hora de encontrar la mejor profundidad para el árbol. Existen diferentes metodologías para encontrar la mejor profundidad, la que se utiliza a continuación es comparar ambas particiones de manera gráfica utilizando como valores del eje y los puntajes y como valores en el eje x la profundidad que se utilizó. Por ende se aíslan dichas características con sus valores para recalcular los árboles de decisión.

Buscando el mejor modelo

Índice de Gini

[44]:
```
MC_Gini =
['Glucosa','Edad','FuncionDiabetesPedigree',
'Insulina','GrosorPiel'] # Mejores Características

X_Gini = X[MC_Gini]
X_Gini = pd.DataFrame(SS_.fit_transform(X_Gini), columns=MC_Gini)
```

[46]:
```
test_size = 1/4
criterion = 'gini'
depth = 15

fun.heuristico_arbol(depth,criterion,X_Gini,y,test_size,rs)
```

```
Puntuación máxima de entrenamiento 1.0 con p = [13, 14]
Puntuación máxima de prueba 0.8061224489795918 con p = [3, 6]
```

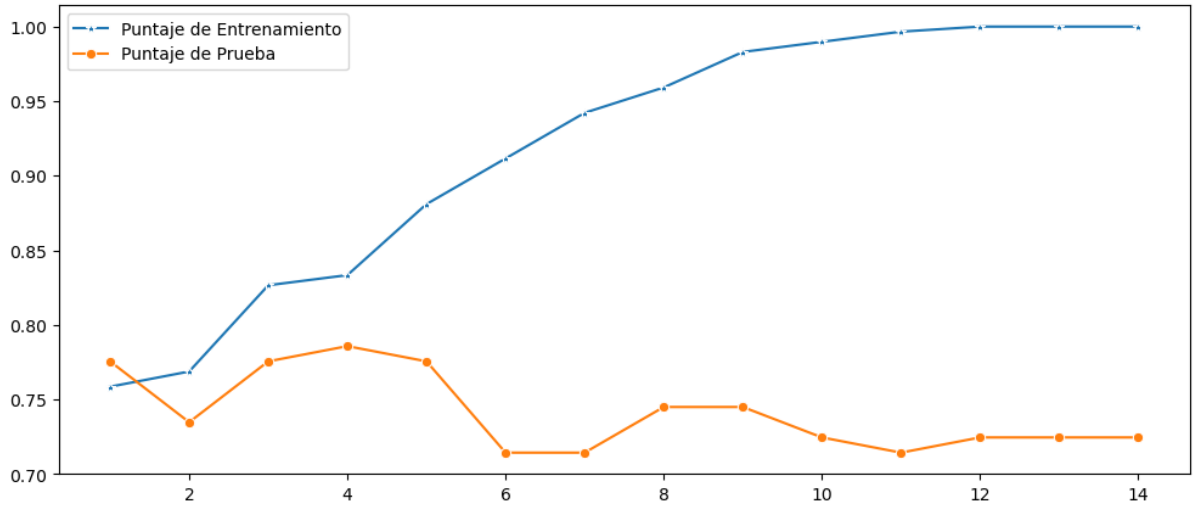

Figura 5.17: Comparación de puntajes de entrenamiento y prueba con Gini.

En la Figura 5.17 ambas curvas se interceptan en el valor de uno y, de ahí, cada puntaje se comporta de manera diferente. Más aún, es notorio que los valores de entrenamiento convergen a hacía 1; y los de prueba a un valor cercano a 0.7. De aquí, el mejor valor para realizar la implementación es el dado en el valor 3 en el conjunto de prueba.

Entropía

[51]:
```
MC_Ent =['Glucosa','Edad',
'FuncionDiabetesPedigree','IMC']    # Mejores Características
X_Ent = X[MC_Ent]
X_Ent = pd.DataFrame(SS_.fit_transform(X_Ent), columns=MC_Ent)
```

[52]:
```
test_size = 1/4
criterion = 'entropy'
depth = 15

fun.heuristico_arbol(depth,criterion,X,y,test_size,rs)
```

```
Puntuación máxima de entrenamiento 1.0 con p = [13, 14]
Puntuación máxima de prueba 0.7959183673469388 con p = [5]
```

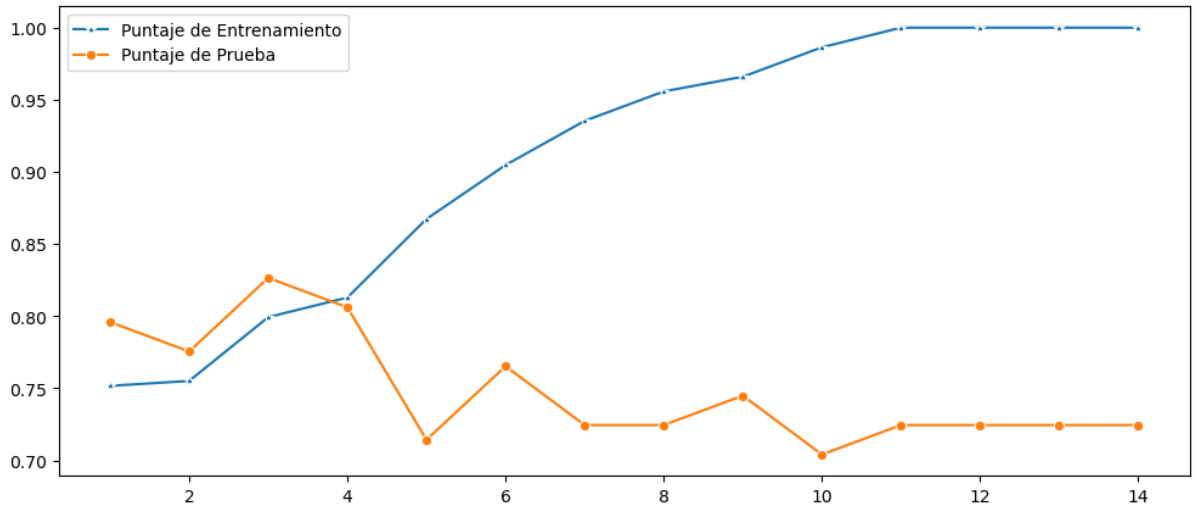

Figura 5.18: Comparación de puntajes de entrenamiento y prueba con Entropía.

En la Figura 5.18 ambas gráficas se interceptan en el valor de cuatro y, de ahí, cada puntaje se comporta de manera diferente. Más aún, es notorio que los valores de entrenamiento convergen hacía 1 (como en la gráfica anterior); y los de prueba a un valor cercano a 0.7 (de la misma forma que la gráfica anterior). De aquí, el mejor valor para realizar la implementación es el dado en el valor 5 en el conjunto de prueba.

Utilizando Entropía para el Modelo Final

[57]:
```
testsize = 1/4
criterion = 'entropy'
depth = 3
rs = 0
etiquetas = X_Ent.columns
CriterioClasif = DecisionTreeClassifier(
criterion = criterion,max_depth=depth,random_state=rs)

fun.Arbol(X_Ent,y,etiquetas,testsize,criterion,depth,rs)
fun.matrx_conf(X_Ent,y,CriterioClasif,testsize,rs)
```

```
Árbol de decisión
Criterio: entropy.
Profundidad: 3
Estadísticos:
Precisión:0.7959
Puntuación Conjunto de Entrenamiento:0.8027
```

```
Puntuación Conjunto de Prueba: 0.7959
Diferencia Absoluta de la Puntuación de los Conjuntos:0.0068
Total de Aciertos: 76
Total de Fracasos: 22
Proporción de Aciertos: 77.55%
```

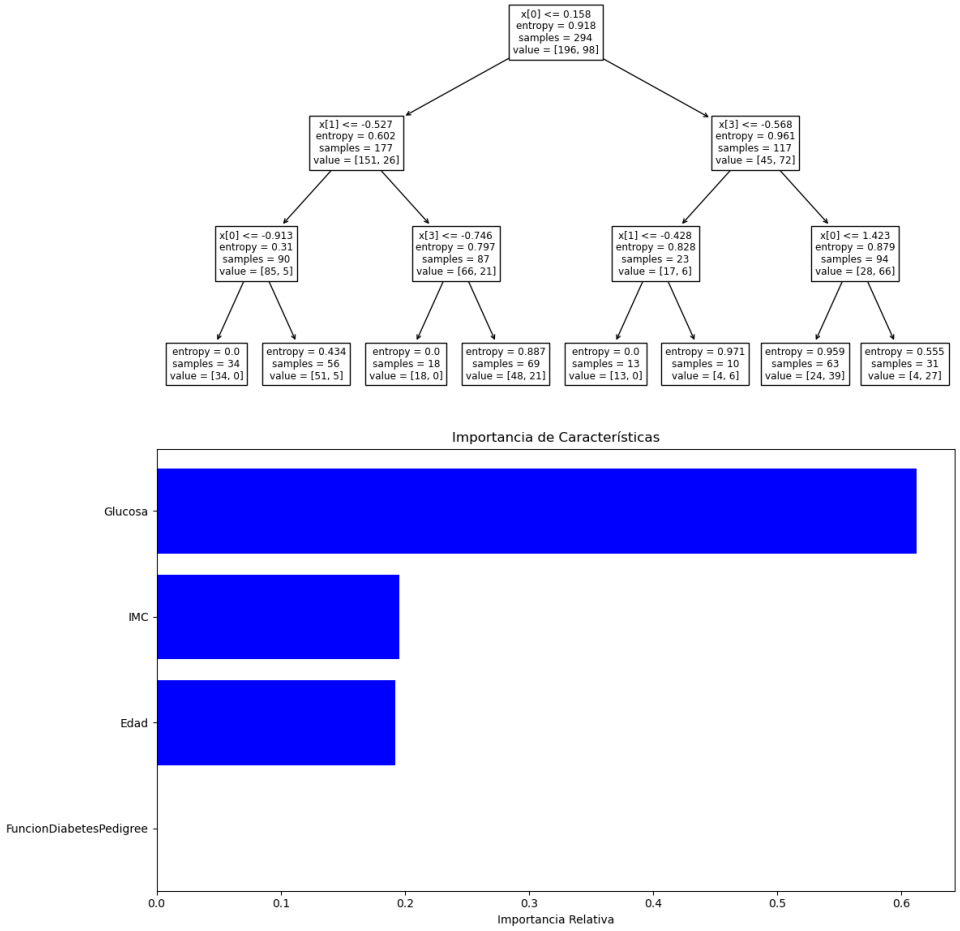

Figura 5.19: Árbol de decisión e importancia relativa generado mediante la entropía.

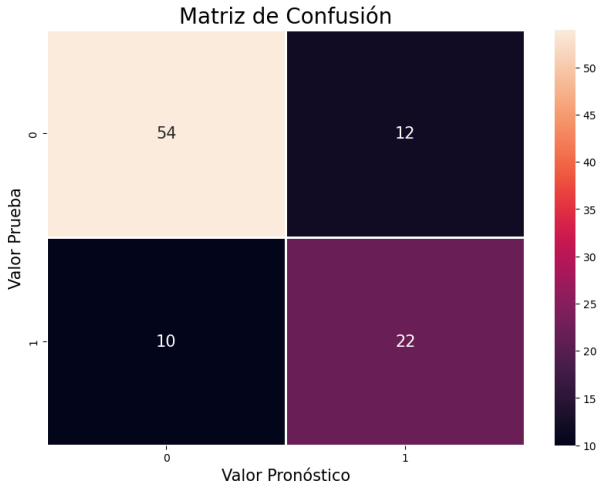

Figura 5.20: Matriz de confusión asociada al modelo de árbol con índice de Gini.

La matriz de confusión mostrada en la Figura 5.21 denota una implementación aceptable. La cual es resumida en la siguiente tabla:

	0	1	Totales
Aciertos	54	12	66
Fracasos	10	22	32

Tabla 5.5: Resumen de los valores de la matriz de confusión.

En la Tabla 5.5 se muestra que, en los verdaderos positivos y los verdaderos negativos, se tiene una predicción buena ya que rebasa el valor de los fallos.

[60]:
```
fun.ROC(X,y,CriterioClasif,testsize,0)
```

```
Sin entrenar: ROC AUC=0.500
Regresión Logística: ROC AUC=0.782
```

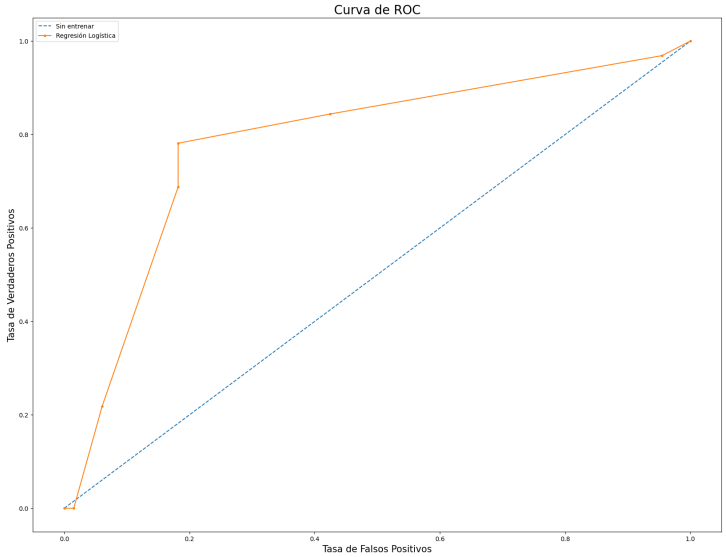

Figura 5.21: Curva de ROC

Por último, es notable que la Figura 5.21 y la Figura 5.16 expresan exactamente la misma curva de ROC.

Conclusión Con una aproximación del 77.5 % se dice que el modelo explica en ese porcentaje la posibilidad de tener diabetes basándose en las características de Glucosa, Edad, Insulina y Función Diabetes Pedigree.

Al final, la importancia relativa muestra que el IMC ya no es sustancial para el modelo. Se podría eliminar de igual forma y volver a hacer el análisis. Pero eso conllevaría hacer múltiples pruebas hasta encontrar el mejor árbol. Para ello es que se ocupa una metodología más avanzada que los árboles de decisión: la metodología de Bosques Aleatorios.

Todas las personas mayores fueron al principio niños,
Aunque pocas de ellas lo recuerdan.

Antoine de Saint-Exupéry, El Principito.

6 Caso de Estudio

Búsqueda y Reconocimiento de Patrones para la Generación de Pronósticos sobre una Base de Datos de la Fundación JUCONI A.C.

6.1. Introducción

El reconocimiento y la búsqueda de patrones para la generación de pronósticos es importante en la toma de decisiones en distintos ámbitos; como el empresarial, económicos e incluso de salud. Sin embargo, el uso de estas tecnologías y conocimientos bien pueden ser aplicados en un ámbito social. Ejemplos de ello se encuentran en las nuevas maneras de implementar la enseñanza mediante análisis de sentimientos, clasificación de enfoques políticos, entre otros.

Como se comenta en el artículo de "Machine learning behaviour: Los agentes de la IA pueden influir en los comportamientos humanos y en los resultados sociales tanto de forma intencionada como no intencionada"[22]. El aprendizaje automático y la inteligencia artificial cada vez permea más en la vida de todas las personas, por eso mismo el uso de estas tecnologías y conocimientos debe de descentralizarse de los ámbitos empresariales, económicos y académicos para dar paso a su uso en pro de la sociedad. En palabras de Carl Sagan [29]: "vivimos en una sociedad totalmente dependiente de la ciencia y la tecnología, en la cual prácticamente nadie sabe nada acerca de la ciencia o la tecnología. Ello constituye una fórmula segura para el desastre". Y, en lo personal, creo que el diversificar las aplicaciones del conocimiento, por ejemplo en el análisis de problemas sociales para ayudar a tener una mejor calidad de vida, abre las puertas a nuevas y diferentes formas de pensamiento, fuera de lo meramente académico.

La Fundación Junto con los Niños (JUCONI) A. C. es una organización de la sociedad civil sin fines de lucro debidamente registrada en México, Estados Unidos y Reino Unido. Fue fundada en la ciudad de Puebla en 1989 cuando Gabriel Benítez, Sarah Thomas de Benítez y Joanna Wright decidieron, con el apoyo del International Children´s Trust, crear una organización. El objetivo de esta fundación es brindar atención profesional a través de programas a niñas y niños en situación de calle, en riesgo o expuestos a la violencia familiar [13]. Esta fundación funciona mediante tres pilares fundamentales:

- Programas JUCONI: Intervenciones directas e intencionadas en atender y mejorar las condiciones de vida de las niñas, niños y adolescentes a través de la construcción

de relaciones basadas en el apego sano y espacios seguros.

- Instituto JUCONI: Investigaciones y consultoría que atiende al público en general, profesionales independientes u organizaciones con interés o conexión al trabajo con niñas, niños y adolescentes que se encuentran en situaciones de vulnerabilidad. Su objetivo es sistematizar el conocimiento que se ha obtenido en el instituto para compartirlo generando así un efecto multiplicador.
- Incidencia JUCONI: Acciones intencionadas a influir en aspectos públicos, políticos y sociales en favor de mejorar las condiciones de vida de las niñas, niños y adolescentes en situaciones de vulnerabilidad.

En este caso de estudio se analiza el pilar de Programas JUCONI y se tiene por meta conocer la influencia de distintas variables dentro de tres Subprogramas de la fundación para generar un modelo predictivo a partir del aprendizaje automático.

El conocer el comportamiento de dichos Subprogramas abre un panorama de análisis y comparación para orientar los bienes y esfuerzos, tanto a áreas de oportunidad, como la mejora de los entornos de Subprogramas.

Un ejemplo de esto es conocer a que Subprograma es más probable que entre un individuo a partir del conocimiento de variables como la escolaridad, edad, ingreso, etc. Lo cual genera una ayuda significativa para la planeación, enfoque y/o replanteamiento de los Subprogramas.

6.1.1. Entendimiento del Problema

Se cuenta con dos conjuntos de datos de la Fundación Junto con los Niños A.C. almacenados en un solo archivo de Excel cada uno con su respectiva hoja de cálculo; etiquetada la primera como "Todos" y la segunda como "Primer-Último". Estas contienen $16,069$ y $5,579$ observaciones respectivamente.

Ambas cuentan con el mismo número de características (o *variables*) salvo "*Primer-Último*", al cual se añade la columna de "TipoReporte" que denota sí los datos del reporte que se ingestaron en la base corresponden al primero que se realizó o al más reciente.

Dichas características son:

1. NRU: Número de registro único.
2. Nombre: Primer nombre de pila.
3. Paterno: Apellido paterno de pila.
4. Materno: Apellido materno de pila.
5. AdultoResponsable: Denota si al individuo se le considera como un adulto responsable.
6. ClasificacionReporte: Tipo de reporte del que se obtuvo la información (N: Niño; J: Joven, A: Adulto).
7. Género: Sexo de nacimiento del individuo.
8. FechaNacimiento: Fecha de nacimiento del individuo.
9. ClaveFamilia: Clave única para identificar a la familia a la que pertenece el individuo.
10. Subprograma: Subprograma al cual está inscrito el individuo.
11. FechaReporte: Fecha en la que se realizó el reporte.
12. EnPrograma: Denota si el individuo está o no inscrito al programa.

13. ParticipacionLaboral: Denota si el individuo labora.
14. Ingreso: Sueldo en pesos que percibe el individuo.
15. Periodicidad: Frecuencia en la que el individuo obtiene el sueldo.
16. AsisteEscuela: Denota si el individuo asiste o no a una institución educativa.
17. SistemaEscolar: Tipo de sistema escolar en el que está inscrito el individuo.
18. UltimanoCompletado: Último grado de estudios que obtuvo el individuo.
19. AsistenciaEscolar: Denota si el individuo asiste o no a la institución educativa.
20. Promedio: Calificación general del último grado de estudios cursado.
21. SabeLeerEscribir: Denota si el individuo es alfabeta o no.
22. TipoReporte: Denota sí, los datos del reporte que se ingestaron a la base, corresponden al primero que se realizó o el más reciente.

El objetivo es plantear hipótesis basadas en la data proporcionada para llegar a generar modelos predictivos que ayuden a la comprensión y toma de decisiones. Para tal objetivo primero la data pasará por un tratamiento en el cual se obtendrá sólo una base de datos homogénea, estructurada y lista para la implementación.

6.1.2. Comprensión de los Datos

En los datos, existen columnas las cuales no son necesarias para el análisis; como el nombre de pila. Además se supone, en primera instancia, una redundancia en las columnas de "ultimocompletado" y "SabeLeerEscribir". Por lo cual se hace una primera exploración de los datos para su preparación.

De manera coloquial, a un conjunto de datos sin tratar se le conoce como *datos sin procesar* (rawdata); y, dado que en este caso se tienen dos rawdatas, se trabajarán a la par.

[1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")

rawdata_TT= pd.read_excel("Reporte de Ingresos.xlsx", sheet_name="Todos")
rawdata_PU= pd.read_excel("Reporte de Ingresos.xlsx",
sheet_name="Primer-último")

rawdata_TT= rawdata_TT.drop(['Nombre','Paterno','Materno'], axis=1)
rawdata_PU= rawdata_PU.drop(['Nombre','Paterno','Materno'], axis=1)

print(rawdata_TT.head(10))
print('==============================================')
print(rawdata_PU.head(10))
```

Todos

NRU	AdultoResponsable	ClasificacionReporte	Género	FechaNacimiento
3202	No	N	Mujer	03/04/2019
3201	No	N	Mujer	30/07/2015
3200	No	N	Mujer	14/06/2011
3199	No	N	Mujer	24/08/2009
3198	Si	A	Mujer	27/08/1993
3158	No	N	Hombre	01/01/2010
3157	No	N	Hombre	01/01/2008
3156	No	N	Hombre	01/01/2006
3155	Si	A	Mujer	01/01/1982
2778	No	N	Hombre	30/05/2015

ClaveFamilia	Subprograma	FechaReporte	enprograma
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
AXOJEJ	Subprograma Niño Trabajador	07/07/2021	si
AXOJEJ	Subprograma Niño Trabajador	07/07/2021	si
AXOJEJ	Subprograma Niño Trabajador	07/07/2021	si
AXOJEJ	Subprograma Niño Trabajador	07/07/2021	si
480QTY	Subprograma Niño Mercado	12/06/2021	si

participacionlaboral	Ingreso	periodicidad	AsisteEscuela	SistemaEscolar
no	0	–	NTES	–
no	0	–	SI	Regular
no	0	–	SI	Regular
si	0	–	SI	Regular
si	1200	Cada	Semana	–
si	0	Cada	Día	SI
si	0	Cada	Día	SI
si	5	Cada	Día	NO
si	150	Cada	Día	–
no	0	–	SI	Regular

ultimanocompletado	AsistenciaEscolar	Promedio	SabeLeerEscribir
Ninguno	–	NaN	–
2PE	Regular	apro	Si
3PE	Regular	apro	No
4P	Irregular	apro	No
NaN	–	Si	–
3PE	Regular	No	–
4P	Regular	Si	–
3P	–	No	–
NaN	–	Si	–
2PE	Regular	No	–

Primer-Último

NRU	AdultoResponsable	ClasificacionReporte	Género	FechaNacimiento
3202	No	N	Mujer	03/04/2019
3202	No	N	Mujer	03/04/2019
3201	No	N	Mujer	30/07/2015
3201	No	N	Mujer	30/07/2015
3200	No	N	Mujer	14/06/2011
3200	No	N	Mujer	14/06/2011
3199	No	N	Mujer	24/08/2009
3199	No	N	Mujer	24/08/2009
3198	Si	A	Mujer	27/08/1993
3198	Si	A	Mujer	27/08/1993

ClaveFamilia	Subprograma	FechaReporte	enprograma
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si
C497EP	Subprograma Niño Trabajador	08/07/2021	si

participacionlaboral	Ingreso	periodicidad	AsisteEscuela	SistemaEscolar
no	0	–	NTES	–
no	0	–	NTES	–
no	0	–	SI	Regular
no	0	–	SI	Regular
no	0	–	SI	Regular
no	0	–	SI	Regular
si	0	–	SI	Regular
si	0	–	SI	Regular
si	1200	Cada	Semana	–
si	1200	Cada	Semana	–

ultimanocompletado	AsistenciaEscolar	Promedio	SabeLeerEscribir	TipoReporte
Ninguno	–	NaN	–	último reporte
Ninguno	–	NaN	–	primer reporte
2PE	Regular	apro	Si	último reporte
2PE	Regular	apro	Si	primer reporte
3PE	Regular	apro	No	último reporte
3PE	Regular	apro	No	primer reporte
4P	Irregular	apro	No	último reporte
4P	Irregular	apro	No	primer reporte
NaN	–	Si	–	último reporte
NaN	–	Si	–	primer reporte

Existe una notable ocurrencia de datos nulos dentro de ambas bases. Para conocer la proporción de estos mismos se calcula, por columna, el número de valores nulos; blancos; y la expresión “–”.

[2]:
```
print("_____________________________________________")
print("CONJUNTO DE DATOS: TODOS")
print("---------------------------------------------")
print("Total de Valores Nulos por Columna")
print("---------------------------------------------")
a = rawdata_TT.isnull().sum()
print(a)
print(f'Total de Valores Nulos: {sum(a)}')
print("---------------------------------------------")
print("Total de Ocurrencias de la Expresión --")
print("---------------------------------------------")
for i in rawdata_TT.columns:
        print(i, list(rawdata_TT[i]).count("--"))
print("---------------------------------------------")
total_1=[]
for i in rawdata_TT.columns:
        total = rawdata_TT[i].isnull().sum() + list(rawdata_TT[i]).count("--")
total_1.append(total)
print(f'Total de Valores de Expresión Nula: {sum(total_1)}')
print("---------------------------------------------")
print(f'Total Acumulado: {sum(a) + sum(total_1)}')
print("_____________________________________________")
print("")
print("_____________________________________________")
print("CONJUNTO DE DATOS: PRIMER-ÚLTIMO")
print("---------------------------------------------")
print("Total de Valores Nulos por Columna")
print("---------------------------------------------")
a = rawdata_PU.isnull().sum()
print(a)
print(f'Total de Valores Nulos: {sum(a)}')
print("---------------------------------------------")
print("Total de Ocurrencias de la Expresión Nula")
print("---------------------------------------------")
for i in rawdata_PU.columns:
        print(i, list(rawdata_PU[i]).count("--"))
print("---------------------------------------------")
total_1=[]
for i in rawdata_PU.columns:
        total = rawdata_PU[i].isnull().sum() + list(rawdata_PU[i]).count("--")
total_1.append(total)
print(f'Total de Valores de Expresión Nula: {sum(total_1)}')
print("---------------------------------------------")
print(f'Total Acumulado: {sum(a) + sum(total_1)}')
print("_____________________________________________")
```

```
---------------------------------------------
CONJUNTO DE DATOS: TODOS
---------------------------------------------
Total de Valores Nulos por Columna
---------------------------------------------
```

```
NRU                        0
AdultoResponsable          0
ClasificacionReporte       0
Género                     0
FechaNacimiento            0
ClaveFamilia               0
Subprograma                0
FechaReporte               0
enprograma                 0
participacionlaboral       0
Ingreso                 6709
periodicidad               0
AsisteEscuela              0
SistemaEscolar             0
ultimanocompletado      6344
AsistenciaEscolar          0
Promedio                5957
SabeLeerEscribir         694
dtype: int64
Total de Valores Nulos: 19704
-----------------------------------------
Total de Ocurrencias de la Expresión --
-----------------------------------------
NRU 0
AdultoResponsable 0
ClasificacionReporte 0
Género 0
FechaNacimiento 0
ClaveFamilia 0
Subprograma 0
FechaReporte 0
enprograma 36
participacionlaboral 2288
Ingreso 0
periodicidad 10030
AsisteEscuela 5727
SistemaEscolar 11073
ultimanocompletado 0
AsistenciaEscolar 8057
Promedio 3
SabeLeerEscribir 4821
-----------------------------------------
Total de Valores de Expresión Nula: 61739
-----------------------------------------
Total Acumulado: 81443
-----------------------------------------

-----------------------------------------
CONJUNTO DE DATOS: PRIMER-ÚLTIMO
-----------------------------------------
Total de Valores Nulos por Columna
```

```
-----------------------------------------
NRU                        0
AdultoResponsable          0
ClasificacionReporte       0
Género                     0
FechaNacimiento            0
ClaveFamilia               0
Subprograma                0
FechaReporte               0
enprograma                 0
participacionlaboral       0
Ingreso                 2080
periodicidad               0
AsisteEscuela              0
SistemaEscolar             0
ultimanocompletado      2324
AsistenciaEscolar          0
Promedio                2290
SabeLeerEscribir         320
TipoReporte                0
dtype: int64
Total de Valores Nulos: 7014
-----------------------------------------
Total de Ocurrencias de la Expresión Nula
-----------------------------------------
NRU 0
AdultoResponsable 0
ClasificacionReporte 0
Género 0
FechaNacimiento 0
ClaveFamilia 0
Subprograma 0
FechaReporte 0
enprograma 2
participacionlaboral 806
Ingreso 0
periodicidad 3450
AsisteEscuela 2010
SistemaEscolar 4082
ultimanocompletado 0
AsistenciaEscolar 2998
Promedio 2
SabeLeerEscribir 1934
TipoReporte 0
-----------------------------------------
Total de Valores de Expresión Nula: 22298
-----------------------------------------
Total Acumulado: 29312
-----------------------------------------
```

Es notable la enorme cantidad de datos nulos dentro del conjunto de Datos. Para el tratamiento de los valores se comienza homologando cada columna y buscando el *valor nulo* dentro del

conjunto de datos de consulta "Todos". Esto debido a que el conjunto de datos "Todos", al eliminar los valores duplicados, tiene la misma cardinalidad que la base de "Primer-Último". Con esto es posible buscar entre algún duplicado los valores faltantes en el conjunto de datos "Primer-Último"; además de que con esto se confirma que el conjunto "Primer-Último" contiene valores únicos.

Para implementar el modelo de manera adecuada, se crea una base de datos que contenga la menor cantidad de nulos y que, además, denote el tipo de reporte de donde se originaron los datos. De modo que la limpieza de la base se centra en "Primer-Último" ya que es la base mejor estructurada de las que se tienen y se utilizarán para rellenar algunos datos faltantes.

6.1.3. Preparación de los Datos

Se comienza por re-etiquetar los encabezados de las columnas de la siguiente manera:

1. NRU: ID_NRU.
2. AdultoResponsable: ADULTO-RESPONSABLE.
3. ClasificacionReporte: CLASIFICACION-REPORTE.
4. Género: SEXO.
5. FechaNacimiento: FNACIMIENTO.
6. ClaveFamilia: ID_CUF.
7. Subprograma: Subprograma.
8. FechaReporte: REPORTE_DATE.
9. Enprograma: ENPROGRAMA.
10. ParticipacionLaboral: PARTICIPACION-LABORAL.
11. Ingreso: INGRESO.
12. Periodicidad: FRECUENCIA_INGRESO.
13. AsisteEscuela: ESCUELA_ASISTE.
14. SistemaEscolar: ESCUELA_SISTEMA-ESCOLAR.
15. UltimanoCompletado: ESCUELA_ESCOLARIDAD.
16. AsistenciaEscolar: ESCUELA_ASISTENCIA.
17. Promedio: ESCUELA_PROMEDIO.
18. SabeLeerEscribir: ESCUELA_ALFABETA.
19. TipoReporte: REPORTE_TIPO

[3]:
```
df_RepIng= pd.DataFrame(rawdata_PU)
df_RepIng.columns = ['ID_NRU','ADULTO-RESPONSABLE',
'REPORTE_CLASIFICACION','SEXO',
'FNACIMIENTO','ID_CUF','Subprograma',
'REPORTE_DATE','ENPROGRAMA','PARTICIPACION-LABORAL','INGRESO',
'INGRESO-FRECUENCIA','ESCUELA_ASISTE','ESCUELA_SISTEMA-ESCOLAR
','ESCUELA_ESCOLARIDAD','ESCUELA_ASISTENCIA',
'ESCUELA_PROMEDIO','ESCUELA_ALFABETA','REPORTE_TIPO']
```

```
print(df_RepIng.head(5))
```

ID_NRU	ADULTO-RESPONSABLE	REPORTE_CLASIFICACION	SEXO
3202	No	N	Mujer
3202	No	N	Mujer
3201	No	N	Mujer
3201	No	N	Mujer
3200	No	N	Mujer

FNACIMIENTO	ID_CUF	Subprograma	REPORTE_DATE
03/04/2019	C497EP	Subprograma Niño Trabajador	08/07/2021
03/04/2019	C497EP	Subprograma Niño Trabajador	08/07/2021
30/07/2015	C497EP	Subprograma Niño Trabajador	08/07/2021
30/07/2015	C497EP	Subprograma Niño Trabajador	08/07/2021
14/06/2011	C497EP	Subprograma Niño Trabajador	08/07/2021

ENPROGRAMA	PARTICIPACION -LABORAL	INGRESO	INGRESO -FRECUENCIA
si	no	0	–
si	no	0	–
si	no	0	–
si	no	0	–
si	no	0	–

ESCUELA_ASISTE	ESCUELA_SISTEMA -ESCOLAR	ESCUELA_ESCOLARIDAD
NTES	–	Ninguno
NTES	–	Ninguno
SI	Regular	2PE
SI	Regular	2PE
SI	Regular	3PE

ESCUELA_ASISTENCIA	ESCUELA_PROMEDIO	ESCUELA_ALFABETA
–	NaN	–
–	NaN	–
Regular	apro	Si
Regular	apro	Si
Regular	apro	No

REPORTE_TIPO
último reporte
primer reporte
último reporte
primer reporte
último reporte

Se prepara la base de datos conociendo los valores únicos y reemplazando el ruido del conjunto.

[4]:
```
p_uniques=
df_RepIng.drop(['ID_NRU','ID_CUF','FNACIMIENTO','REPORTE_DATE'], axis=1)
print('VALORES ÚNICOS POR COLUMNA')
print('============================================================')
for cols in p_uniques.columns:
        print(f'{cols}: {df_RepIng[cols].unique()}')
```

```
VALORES ÚNICOS POR COLUMNA
================================================================================
ADULTO-RESPONSABLE: ['No' 'Si']

REPORTE_CLASIFICACION: ['N' 'A' 'J']

SEXO: ['Mujer' 'Hombre']

Subprograma: ['Subprograma Niño Trabajador' 'Subprograma Niño Mercado'
'Subprograma Niño Calle']

ENPROGRAMA: ['si' 'no' '--']

PARTICIPACION-LABORAL: ['no' 'si' '--']

INGRESO: [0.000e+00 1.200e+03 5.000e+00 1.500e+02 1.000e+03 8.000e+02 3.000e+03
5.000e+02 2.500e+02 2.200e+02 5.500e+02 6.000e+02 2.000e+02 1.800e+03
1.000e+02 1.400e+03 7.000e+02 4.000e+02 2.000e+03 3.000e+01 5.000e+01
8.000e+01 9.000e+02 1.300e+03 3.000e+02 1.600e+03 1.500e+03 3.500e+03
1.900e+03 7.500e+02 1.600e+02 9.500e+02 8.500e+02 3.360e+03 1.050e+03
1.800e+02 1.200e+02 1.100e+03 2.500e+03 1.700e+03 4.000e+03 1.000e+04
3.500e+02 2.300e+02 9.000e+01 8.000e+00 1.300e+02 6.000e+03 7.000e+03
6.750e+02 5.000e+03 7.000e+01 4.500e+02 9.700e+02 7.800e+02 3.600e+03
3.900e+02 4.500e+03 4.000e+01 7.470e+02       nan 6.500e+02 5.800e+03
1.200e+04 3.700e+03 3.200e+03 6.000e+01 2.700e+03 1.750e+03 1.700e+02
5.100e+02 1.250e+03 1.030e+03 3.500e+01 1.650e+02 2.644e+03 1.750e+02
4.500e+01 1.100e+02 4.200e+01 4.300e+01 1.400e+02 3.800e+01 9.300e+01
8.500e+01]

INGRESO-FRECUENCIA: ['--' 'Cada Semana' 'Cada Día' 'Cada Mes' 'Cada Quincena'
'Cada Semestre']

ESCUELA_ASISTE: ['NTES' 'SI' '--' 'NO']

ESCUELA_SISTEMA-ESCOLAR: ['--' 'Regular' 'Abierto']

ESCUELA_ESCOLARIDAD: ['Ninguno' '2PE' '3PE' '4P' nan '3P' '3S' '2P' '5P' '1S'
'1P' '6P' '1PE'
'3PREPA' '2PREPA' '2S' '1PREPA' 'Ninguno pero es alfabeta' 'Técnica'
'Licenciatura']

ESCUELA_ASISTENCIA: ['--' 'Regular' 'Irregular']

ESCUELA_PROMEDIO: [' ' 'apro' ' N/A' 'APROB' ' apro' 'aprob' 8 0 7.6 6 8.3
'Aprob' 7.5 7
' NA' 7.7 nan 'apr' 'no' ' Apro' 8.9 9 8.5 9.5 7.2 ' APRO' 8.6 8.4 8.2
' Apr' 7.4 6.4 ' n/a' 6.5 6.2 ' N/P' 'Repro' 6.7 8.8 7.9 6.8 'APROV' 6.3
'aprov' 'Promo' 8.7 7.8 'Desconocido' 'Aprov' 'APRO' 'aprobada' 'Aprobo'
'aprobo' 'aprobado' ' Aprb' 8.1 'Aprobó' 7.3 ' --' 9.8 9.4 'No tiene'
'ABROB' ' prom' 'AP' 9.3 'Aprobado' 6.9 88 68 'N.A' 87 'N. A' 'Aprobada'
```

```
9.2 'aprobó' 7.1 5 9.6 6.6 6.1 9.1 '--' 10]

ESCUELA_ALFABETA: [nan 'Si' 'No' '--']

REPORTE_TIPO: ['último reporte' 'primer reporte']
```

El ruido dentro de las columnas se arregla definiendo los valores de cada columnas como sigue:

- ADULTO-RESPONSABLE: {'SI','NO'}.
- REPORTE_CLASIFICACION: {'N','A','J'}.
- SEXO: {'M','F'}.
- Subprograma: {'NIÑO CALLE', 'NIÑO MERCADO', 'NIÑO TRABAJADOR'}
- ENPROGRAMA: {'SI','NO'}
- PARTICIPACION-LABORAL: {'SI','NO'}
- INGRESO-FRECUENCIA: {'DIARIO','SEMANAL','QUINCENAL','MENSUAL','SEMESTRAL'}
- ESCUELA_ASISTE: {'SI','NO'}
- ESCUELA_SISTEMA-ESCOLAR: {'REGULAR','ABIERTO'}
- ESCUELA_ASISTENCIA: {'REGULAR','IRREGULAR'}
- ESCUELA_ALFABETA: {'SI','NO'}
- REPORTE_TIPO: {'R1','R2'}[1]

Los valores de las columnas *ESCUELA_ESCOLARIDAD* y *ESCUELA_PROMEDIO* se tratan de manera más específica y los valores que denotan *nulos* se expresan, dentro de la base como: '–', '– ', nan, vacío y blanco.

ADULTO-RESPONSABLE, SEXO, SUBPROGRAMA, ENPROGRAMA, PARTICIPACION-LABORAL, INGRESO-FRECUENCIA, ESCUELA_ASISTE, ESCUELA_SISTEMA-ESCOLAR, ESCUELA_ASISTENCIA, ESCUELA_ALFABETA, REPORTE_TIPO

[5]:
```
df_RepIng= df_RepIng.replace(['Si','No'],['SI','NO'])
df_RepIng= df_RepIng.replace(['si','no'],['SI','NO'])
df_RepIng= df_RepIng.replace(['SI','NO', 'NTES'],['SI','NO','NO'])
df_RepIng= df_RepIng.replace(['Mujer','Hombre'],['F','M'])
df_RepIng= df_RepIng.replace([
'Subprograma Niño Mercado',
'Subprograma Niño Trabajador',
'Subprograma Niño Calle'],['Niño Mercado','Niño Trabajador','Niño Calle'])
df_RepIng= df_RepIng.replace(['Cada Día','Cada Semana','Cada Mes',
'Cada Quincena',
'Cada Semestre'],['DIARIO','SEMANAL','MENSUAL','QUINCENAL','SEMESTRAL'])
df_RepIng= df_RepIng.replace(['Abierto','Regular'],['ABIERTO','REGULAR'])
df_RepIng= df_RepIng.replace(['Irregular'],['IRREGULAR'])
df_RepIng= df_RepIng.replace(['primer reporte','último reporte'],['R1','R2'])
df_RepIng= df_RepIng.replace(['--','', ' ',' --', 'nan', 'NaN'],
[np.nan,np.nan,np.nan,np.nan,np.nan, np.nan])

p_uniques= df_RepIng.drop(['ID_NRU','ID_CUF',
'FNACIMIENTO','REPORTE_DATE'], axis=1)
print('VALORES ÚNICOS POR COLUMNA')
```

[1] R1 denota el primer reporte y R2 denota el último reporte.

```
for cols in p_uniques.columns:
        print(f'{cols}: {df_RepIng[cols].unique()}')
```

```
VALORES ÚNICOS POR COLUMNA

ADULTO-RESPONSABLE: ['NO' 'SI']

REPORTE_CLASIFICACION: ['N' 'A' 'J']

SEXO: ['F' 'M']

Subprograma: ['Niño Trabajador' 'Niño Mercado' 'Niño Calle']

ENPROGRAMA: ['SI' 'NO' nan]

PARTICIPACION-LABORAL: ['NO' 'SI' nan]

INGRESO: [0.000e+00 1.200e+03 5.000e+00 1.500e+02 1.000e+03 8.000e+02 3.000e+03
5.000e+02 2.500e+02 2.200e+02 5.500e+02 6.000e+02 2.000e+02 1.800e+03
1.000e+02 1.400e+03 7.000e+02 4.000e+02 2.000e+03 3.000e+01 5.000e+01
8.000e+01 9.000e+02 1.300e+03 3.000e+02 1.600e+03 1.500e+03 3.500e+03
1.900e+03 7.500e+02 1.600e+02 9.500e+02 8.500e+02 3.360e+03 1.050e+03
1.800e+02 1.200e+02 1.100e+03 2.500e+03 1.700e+03 4.000e+03 1.000e+04
3.500e+02 2.300e+02 9.000e+01 8.000e+00 1.300e+02 6.000e+03 7.000e+03
6.750e+02 5.000e+03 7.000e+01 4.500e+02 9.700e+02 7.800e+02 3.600e+03
3.900e+02 4.500e+03 4.000e+01 7.470e+02       nan 6.500e+02 5.800e+03
1.200e+04 3.700e+03 3.200e+03 6.000e+01 2.700e+03 1.750e+03 1.700e+02
5.100e+02 1.250e+03 1.030e+03 3.500e+01 1.650e+02 2.644e+03 1.750e+02
4.500e+01 1.100e+02 4.200e+01 4.300e+01 1.400e+02 3.800e+01 9.300e+01
8.500e+01]

INGRESO-FRECUENCIA: [nan 'SEMANAL' 'DIARIO' 'MENSUAL' 'QUINCENAL' 'SEMESTRAL']

ESCUELA_ASISTE: ['NO' 'SI' nan]

ESCUELA_SISTEMA-ESCOLAR: [nan 'REGULAR' 'ABIERTO']

ESCUELA_ESCOLARIDAD: ['Ninguno' '2PE' '3PE' '4P' nan '3P' '3S' '2P' '5P' '1S'
'1P' '6P' '1PE'
'3PREPA' '2PREPA' '2S' '1PREPA' 'Ninguno pero es alfabeta' 'Técnica'
'Licenciatura']

ESCUELA_ASISTENCIA: [nan 'REGULAR' 'IRREGULAR']

ESCUELA_PROMEDIO: [nan 'apro' ' N/A' 'APROB' ' apro' 'aprob' 8 0 7.6 6 8.3
'Aprob' 7.5 7
' NA' 7.7 'apr' 'NO' ' Apro' 8.9 9 8.5 9.5 7.2 ' APRO' 8.6 8.4 8.2 ' Apr'
7.4 6.4 ' n/a' 6.5 6.2 ' N/P' 'Repro' 6.7 8.8 7.9 6.8 'APROV' 6.3 'aprov'
'Promo' 8.7 7.8 'Desconocido' 'Aprov' 'APRO' 'aprobada' 'Aprobo' 'aprobo'
'aprobado' ' Aprb' 8.1 'Aprobó' 7.3 9.8 9.4 'No tiene' 'ABROB' ' prom'
'AP' 9.3 'Aprobado' 6.9 88 68 'N.A' 87 'N. A' 'Aprobada' 9.2 'aprobó' 7.1
```

```
5 9.6 6.6 6.1 9.1 10]

ESCUELA_ALFABETA: [nan 'SI' 'NO']

REPORTE_TIPO: ['R2' 'R1']
```

Ahora, para las columnas de 'FNACIMIENTO', 'INGRESO', 'INGRESO_FRECUENCIA' 'ESCUELA_PROMEDIO' y 'ESCUELA_ESCOLARIDAD' se tratan por separado de la siguiente forma:

FECHA DE NACIMIENTO (EDAD)

A partir de conocer la fecha de nacimiento del individuo, se procede a calcular la edad hasta la fecha más actual de los reportes; la cual es el 08/07/2021. Por lo cual se adhiere una nueva columna a las bases llamada 'EDAD'.

Para trabajar con fechas de manera adecuada con Python, es necesario hacer uso del entorno 'datetime'.

[6]:
```
from datetime import datetime
from dateutil.relativedelta import relativedelta

fecha_u = datetime.strptime("8/7/2021", "%d/%m/%Y")
EDAD_=[]
for ed in df_RepIng['FNACIMIENTO']:
        edad = relativedelta(fecha_u, ed)
EDAD_.append(edad.years)

df_RepIng['EDAD']=EDAD_
print(f' Primeros 5 valores de la columna EDAD: {EDAD_[0:5]}')
```

```
Primeros 5 valores de la columna EDAD: [2, 2, 5, 5, 10]
```

INGRESO y FRECUENCIA (INGRESO QUINCENAL)

Se crea una columna llamada 'INGRESO_QUINCENAL', a partir de las columnas de 'INGRESO' y de 'INGRESO_FRECUENCIA' de la siguiente manera:

[7]:
```
QUINCENAL=[]

for ing in range(0,len(df_RepIng['INGRESO'])):
        if df_RepIng['INGRESO-FRECUENCIA'][ing]=='DIARIO':
                QUINCENAL.append(df_RepIng['INGRESO'][ing]*15)
        elif df_RepIng['INGRESO-FRECUENCIA'][ing]=='SEMANAL':
                QUINCENAL.append(df_RepIng['INGRESO'][ing]*2)
        elif df_RepIng['INGRESO-FRECUENCIA'][ing]=='QUINCENAL':
                QUINCENAL.append(df_RepIng['INGRESO'][ing])
        elif df_RepIng['INGRESO-FRECUENCIA'][ing]=='MENSUAL':
                QUINCENAL.append(df_RepIng['INGRESO'][ing]/2)
        elif df_RepIng['INGRESO-FRECUENCIA'][ing]=='SEMESTRAL':
                QUINCENAL.append(df_RepIng['INGRESO'][ing]/12)
```

```
        else: QUINCENAL.append(0)

df_RepIng['INGRESO_QUINCENAL']=QUINCENAL
```

PROMEDIO (PROMOCIÓN)

Para transformar de manera adecuada la columna de 'ESCUELA_PROMEDIO', se transforman todos los tipos a dos en concreto:

- Promovido: Individuos que denoten un valor que se considere como promovido (de manera numérica, mayor o igual a seis).
- No Promovidos: Individuos que denoten un valor que se considere como no promovido (de manera numérica, menor a seis).

El diccionario que se genera para conocer que valor entra en cual categoría se construye con los tipos únicos y acomodando en la categoría que le corresponde.

- La información que se tiene para que los individuos se clasifiquen como *Promovidos* son denotados con los siguientes registros:

  {'Aprobado', 'Aprobo', 'Aprobó', 'aprobó', 'aprob', 'Aprob', 'Aprobada', 'APROB', 'APRO', 'AP', ' Apro', 'ABROB', ' prom', ' APRO', 'Aprov', 'aprobo', ' Aprb', 'aprobado', 'aprobada', 'aprov', 'apro', ' apro', 'APROV', datetime.time(8,4), 'Promo', ' Apr', 'apr', valores iguales o mayores que el seis }.

- Mientras que los *No Promovidos* se denotan mediante:

  {'Repro',' N/P', ' NA', 'N.A', 'N. A', valores abajo del seis, 'NO' }.

- Y, para finalizar, los valores *Nulos* se entienden de la siguiente forma:

  {'Desconocido', 'NaN', nan, 'No tiene'}

[8]:
```
df_RepIng['ESCUELA_PROMEDIO'] = df_RepIng['ESCUELA_PROMEDIO'].replace(
['Aprobado', 'Aprobo',
'Aprobó', 'aprobó',
'aprob', 'Aprob',
'Aprobada', 'APROB',
'APRO', 'AP',
' Apro', 'ABROB',
' prom', ' APRO',
'Aprov', 'aprobo', ' Aprb',
'aprobado', 'aprobada',
'aprov', 'apro',
' apro', 'APROV', 'Promo',
' Apr', 'apr'],
['PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO', 'PROMOVIDO',
```

```
'PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO', 'PROMOVIDO',
'PROMOVIDO','PROMOVIDO']
)

df_RepIng['ESCUELA_PROMEDIO'] = df_RepIng['ESCUELA_PROMEDIO'].replace(
['N/A', 'NO', 'Repro', 'Desconocido','No tiene','N.A',
'N. A', ' N/A',
'N/P', ' N/P',' n/a'],
['NO PROMOVIDO',
'NO PROMOVIDO','NO PROMOVIDO',
'NO PROMOVIDO','NO PROMOVIDO',
'NO PROMOVIDO',
'NO PROMOVIDO','NO PROMOVIDO',
'NO PROMOVIDO','NO PROMOVIDO',
'NO PROMOVIDO']
)

df_RepIng['ESCUELA_PROMEDIO']=df_RepIng['ESCUELA_PROMEDIO'].replace(['Repro',
' N/P',' NA', 'N.A','N. A',' N/A', ' n/a'],['NO PROMOVIDO',
'NO PROMOVIDO','NO PROMOVIDO',
'NO PROMOVIDO', 'NO PROMOVIDO',
'NO PROMOVIDO','NO PROMOVIDO'])

df_RepIng['ESCUELA_PROMEDIO']=
df_RepIng['ESCUELA_PROMEDIO'].replace(['Desconocido',
'NaN',
'No tiene',
'NO'],
[np.nan,
np.nan,np.nan,
np.nan])

pro=[]
for i in df_RepIng['ESCUELA_PROMEDIO']:
        if type(i)==type('a'):
                pro.append(i)
        elif type(i)!=type('a'):
                if i < 6:
                        pro.append('NO PROMOVIDO')
                else: pro.append('PROMOVIDO')


df_RepIng['ESCUELA_PROMOCION']=pro
```

ESCUELA_ESCOLARIDAD

Para concluir con el homologado de datos, la columna 'ESCUELA_ESCOLARIDAD' cambia con respecto al siguiente diccionario:

- Kinder: K={'1PE','2PE','3PE'}
- Primeros Tres Años de Primaria: P_1={'3P','2P','1P'}
- Últimos Tres Años de Primaria: P_2={'6P','5P','4P'}
- Secundaria: S={'1S','2S','3S'}
- Preparatoria: PP={'3PREPA','2PREPA','1PREPA'}
- Licenciatura: L={'Licenciatura'}
- Técnica: T={'Técnica'}
- Alfabeta: A={'Ninguno pero es alfabeta'}
- Ninguno: N={'Ninguno'}

[9]:
```
df_RepIng['ESCUELA_ESCOLARIDAD']=
df_RepIng['ESCUELA_ESCOLARIDAD'].replace(['1PE','2PE','3PE'],['K','K','K'])
df_RepIng['ESCUELA_ESCOLARIDAD']=
df_RepIng['ESCUELA_ESCOLARIDAD'].replace(['3P','2P','1P'],['P_1','P_1','P_1'])
df_RepIng['ESCUELA_ESCOLARIDAD']=
df_RepIng['ESCUELA_ESCOLARIDAD'].replace(['6P','5P','4P'],['P_2','P_2','P_2'])
df_RepIng['ESCUELA_ESCOLARIDAD']=
df_RepIng['ESCUELA_ESCOLARIDAD'].replace(['1S','2S','3S'],['S','S','S'])
df_RepIng['ESCUELA_ESCOLARIDAD']=
df_RepIng['ESCUELA_ESCOLARIDAD'].replace(
['3PREPA','2PREPA','1PREPA'],['PP','PP','PP']
)
df_RepIng['ESCUELA_ESCOLARIDAD']=
df_RepIng['ESCUELA_ESCOLARIDAD'].replace(['Licenciatura'],['L'])
df_RepIng['ESCUELA_ESCOLARIDAD']=
df_RepIng['ESCUELA_ESCOLARIDAD'].replace(['Técnica'],['T'])
df_RepIng['ESCUELA_ESCOLARIDAD']=
df_RepIng['ESCUELA_ESCOLARIDAD'].replace(['Ninguno pero es alfabeta'],['A'])
df_RepIng['ESCUELA_ESCOLARIDAD']=
df_RepIng['ESCUELA_ESCOLARIDAD'].replace(['Ninguno'],['N'])
```

Con esto queda limpia la base del ruido que contenía. El siguiente paso es encontrar las redundancias de la base; por ejemplo las columnas de 'ESCUELA_ESCOLARIDAD' y 'ESCUELA_ALFABETA'. Esto debido a que la columna de 'ESCUELA_ALFABETA', apriori, resulta redundante; ya que, en la columna de 'ESCUELA_ESCOLARIDAD', se denota si el individuo no tiene una escolaridad previa pero es alfabeta con el símbolo 'A'. De modo que cualquier escolaridad, excepto 'N', 'K' y nan, deberían de tener un valor de 'SI' en la columna de 'ESCUELA_ALFABETA'. Pero, al notar la asistencia del individuo, el alfabetísmo se ve mermado con el valor de irregularidad. Aquí es donde se discierne si la columna es significante para el modelo.

Como es de interés conocer si el individuo tiene, o no, un grado de estudios; el denotar, de manera particular si es alfabeta o no, se considera redundante. Por consiguiente se elimina esta columna del modelo junto con las que se usaron para calcular las nuevas columnas. Dando como resultado la siguiente base:

[10]:
```
df_RepIng=
df_RepIng.drop(['FNACIMIENTO','INGRESO',
'INGRESO-FRECUENCIA','ESCUELA_PROMEDIO',
'ESCUELA_ALFABETA'],axis=1)
```

Para el rellenado de variables, se utiliza la herramienta Orange for Data Mining como sigue:

[11]:
```
df_RepIng.to_excel('df_RepIng.xlsx')
```

Orange for Data Mining

Es un programa orientado a los procesos para implementar modelos de aprendizaje automático. Contiene herramientas para la importación, visualización, transformación, implementación y evaluación de modelos de aprendizaje automático con un flujo de trabajo intuitivo del tipo *point-n-click* en su mayoría. Este programa se utiliza para la exploración y la transformación de datos como sigue:

1. Se importan los datos en el formato que se almacenaron (excel, csv, table, etc.). Inciso i de la Figura 6.1.
2. Se exploran los datos mediante distintos diagramas para plantear la mejor manera de rellenar los datos nulos. Incisos ii, iii, iv, v de la Figura 6.1.
3. Se rellenan los datos mediante un modelo de árboles simple determinado por el programa. Inciso vi de la Figura 6.1.
4. Se generan nuevos diagramas para compararlos con los primeros y encontrar errores o inconsistencias. Incisos $vii, viii, ix, x$ de la Figura 6.1.
5. Se exporta la nueva base de datos. Inciso xi de la Figura 6.1.

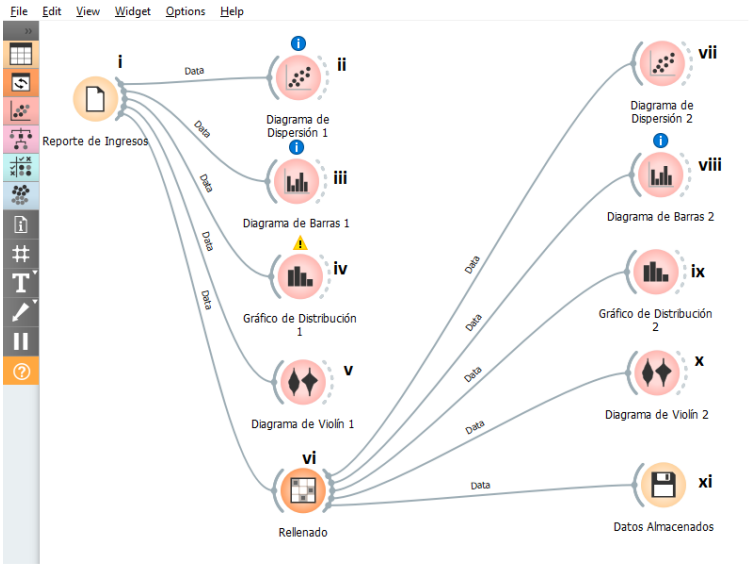

Figura 6.1: Flujo de trabajo de Orange.

Se eliminan del modelo a los individuos menores a los 16 años de edad, la razón deriva de la dispersión generada en los individuos de 15 años hacía abajo, la cual puede ocasionar errores al momento de hacer las predicciones.

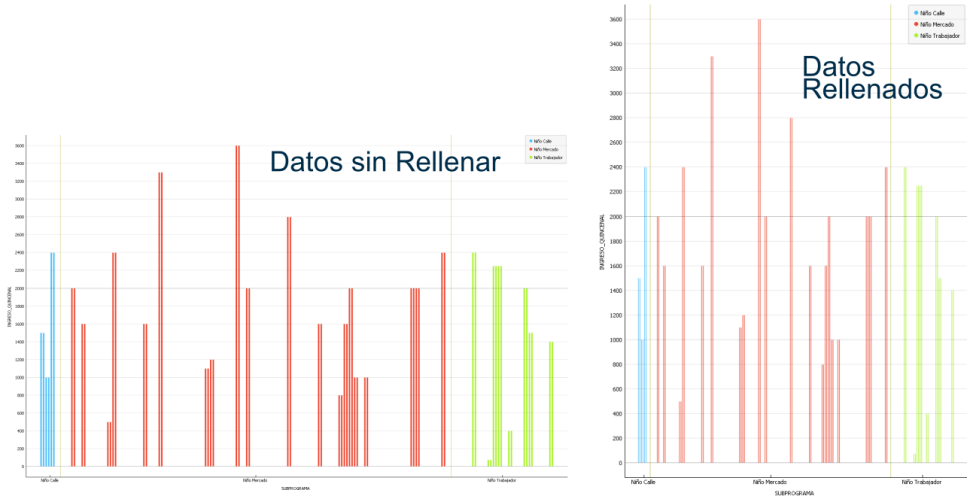

Figura 6.2: Comparación de datos originales v.s. rellenados en gráfico de barras.

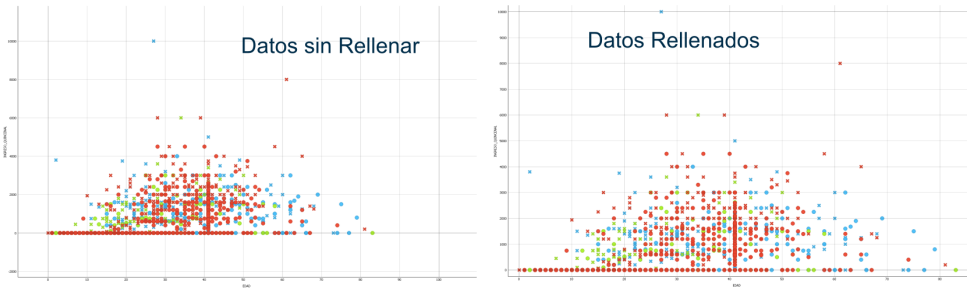

Figura 6.3: Comparación de datos originales v.s. rellenados en gráfico de dispersión.

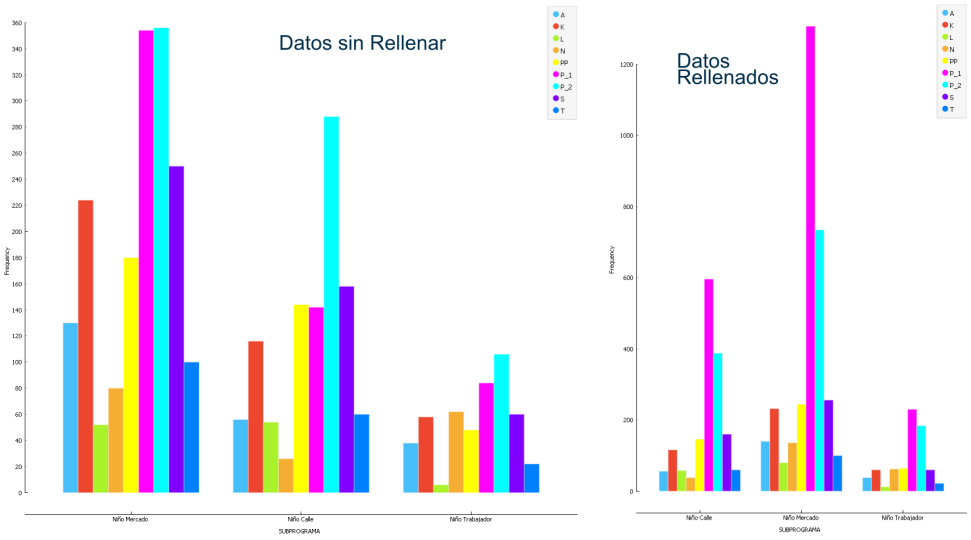

Figura 6.4: Comparación de datos originales v.s. rellenados en gráfico de distribución.

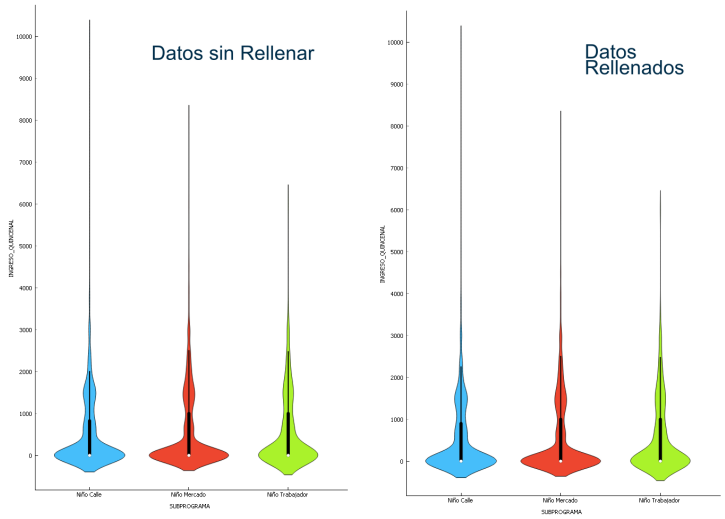

Figura 6.5: Comparación de datos originales v.s. rellenados en gráfico de violines.

En la Figura 6.5, lo que se busca es llegar a una distribución similar a los valores originales. Pueden causar dudas sobre la mediana en cero, pero esto es consecuencia de que los valores son dicótomos. Lo que indica es la existencia de muchos ceros en la base.

Generada entonces la nueva data se procede a cargarla de nuevo dentro de Jupyter Notebook, tomar los valores del reporte más reciente, eliminar duplicados, re-indexar los valores y comenzar con la etapa de modelado.

[12]:
```
Data_r= pd.read_excel('Reporte de Ingresos Consolidado ORNG.xlsx')
Data_r= Data_r.drop(['Feature 1'], axis=1)
r2= Data_r['REPORTE_TIPO']=="R2"
df= Data_r[r2]
df= df.drop_duplicates()
df.index= range(len(df))
df= df.drop(['REPORTE_TIPO'],axis=1)
df
```

ID_NRU	ADULTO-RESPONSABLE	REPORTE_CLASIFICACION
3198	SI	A
3156	NO	N
3155	SI	A
2776	SI	A
2783	SI	A
...	...	...
570	SI	A
968	NO	N
1001	SI	A
963	NO	N
1011	SI	A

SEXO	SUBPROGRAMA	REPORTE_DATE
F	Niño Trabajador	08/07/2021
M	Niño Trabajador	07/07/2021
F	Niño Trabajador	07/07/2021
M	Niño Mercado	12/06/2021
M	Niño Trabajador	06/06/2021
...	...	...
F	Niño Mercado	01/08/1999
M	Niño Trabajador	01/06/1999
F	Niño Trabajador	12/03/1999
M	Niño Trabajador	12/03/1999
F	Niño Trabajador	01/03/1999

ENPROGRAMA	PARTICIPACION-LABORAL
SI	SI
SI	SI
SI	SI
SI	SI
SI	SI
...	...
SI	SI
SI	SI
SI	SI
SI	SI
SI	SI

ESCUELA_ASISTE	ESCUELA_SISTEMA -ESCOLAR	ESCUELA_ESCOLARIDAD
NO	REGULAR	PP
NO	REGULAR	P_1
NO	ABIERTO	P_1
NO	ABIERTO	P_1
NO	ABIERTO	P_1
...	...	...
NO	ABIERTO	P_1
SI	ABIERTO	P_2
NO	ABIERTO	P_1
SI	ABIERTO	P_1
NO	ABIERTO	P_1

ESCUELA_ASISTENCIA	EDAD	INGRESO_QUINCENAL
IRREGULAR	27	2400
IRREGULAR	15	75
REGULAR	39	2250
REGULAR	45	2000
REGULAR	48	2250
...	...	...
REGULAR	41	1395
REGULAR	32	600
REGULAR	41	1275
REGULAR	31	600
REGULAR	41	570

ESCUELA_PROMOCION	ID_CUF
PROMOVIDO	C497EP
PROMOVIDO	AXOJEJ
PROMOVIDO	AXOJEJ
PROMOVIDO	480QTY
PROMOVIDO	M3CONC
...	...
PROMOVIDO	EAV1Y7
PROMOVIDO	AL1PWT
PROMOVIDO	ZQ3T1M
PROMOVIDO	ZQ3T1M
PROMOVIDO	J9CRYK

En esta etapa se implementan los modelos de aprendizaje automático para los que el modelo es candidato y se evaluan entre sí.

Antes de implementar un modelo como tal, es necesario transformar los datos categóricos a númericos mediante variables dummies. Aquí hay que tener especial cuidado en la interpretación de los valores.

Si se tienen características con más de 2 valores categóricos para hacerse dummie, hay que identificarlas y segmentarlas de los valores categóricos binarios. Dándo así dos particiones de la base total.

La primera partición que contiene los valores binarios desde origen se le etiqueta como “df_b”; mientras que la segunda partición recibe la etiqueta de “df_nb”. Al segmentarlas y tratarlas por separado se evita que la función que calcula los dummies realice transformaciones redundantes y para que la lógica de la base se conserve con la transformación.

La partición df_b lleva por columnas:

- ADULTO-RESPONSABLE.
- SEXO.
- ENPROGRAMA.
- PARTICIPACION-LABORAL.
- ESCUELA_ASISTE.
- ESCUELA_SISTEMA-ESCOLAR.
- ESCUELA_ASISTENCIA.

- ESCUELA_PROMOCION.

La partición df_nb contiene el resto de columnas exceptuando las que contienen valores de idenficación únicos (ID's) y las que contienen valores tipo fecha; los cuales se almacenan en un dataframe llamado "extras". Estas tres particiones forman la base de datos entera.

[13]:
```
extras= df[['REPORTE_DATE', 'ID_CUF', 'ID_NRU']]
df_b=df[['ADULTO-RESPONSABLE','SEXO',
'ENPROGRAMA','PARTICIPACION-LABORAL',
'ESCUELA_ASISTE','ESCUELA_SISTEMA-ESCOLAR',
'ESCUELA_ASISTENCIA','ESCUELA_PROMOCION']]
df_nb=df.drop(['ADULTO-RESPONSABLE','SEXO',
'ENPROGRAMA','PARTICIPACION-LABORAL',
'ESCUELA_ASISTE','ESCUELA_SISTEMA-ESCOLAR',
'ESCUELA_ASISTENCIA','ESCUELA_PROMOCION',
'ID_CUF','REPORTE_DATE', 'ID_NRU'], axis=1)
```

Segmentados los datos, se tratan por separado de la siguiente manera:

[14]:
```
df_b['ADULTO-RESPONSABLE']=
df_b['ADULTO-RESPONSABLE'].replace({"SI": 1, "NO": 0});
df_b['SEXO']=df_b['SEXO'].replace({"F": 1, "M": 0});
df_b['ENPROGRAMA']=df_b['ENPROGRAMA'].replace({"SI": 1, "NO": 0});
df_b['PARTICIPACION-LABORAL']=
df_b['PARTICIPACION-LABORAL'].replace({"SI": 1, "NO": 0});
df_b['ESCUELA_ASISTE']=df_b['ESCUELA_ASISTE'].replace({"SI": 1, "NO": 0});
df_b['ESCUELA_SISTEMA-ESCOLAR']=
df_b['ESCUELA_SISTEMA-ESCOLAR'].replace({"REGULAR": 1, "ABIERTO": 0});
df_b['ESCUELA_ASISTENCIA']=
df_b['ESCUELA_ASISTENCIA'].replace({"REGULAR": 1, "IRREGULAR": 0});
df_b['ESCUELA_PROMOCION']=
df_b['ESCUELA_PROMOCION'].replace({"PROMOVIDO": 1, "NO PROMOVIDO": 0});
df_nb_dum=pd.get_dummies(df_nb)
df= pd.concat([df_b, df_nb_dum, extras], axis=1)
```

Con esto se construye la matriz de correlación como sigue:

Matriz de Correlación

[15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
df_corr= df.drop(['REPORTE_DATE','ID_CUF','ID_NRU'],axis=1)

corr_df = df_corr.corr()

corr_df.style.background_gradient (cmap = 'coolwarm')
```

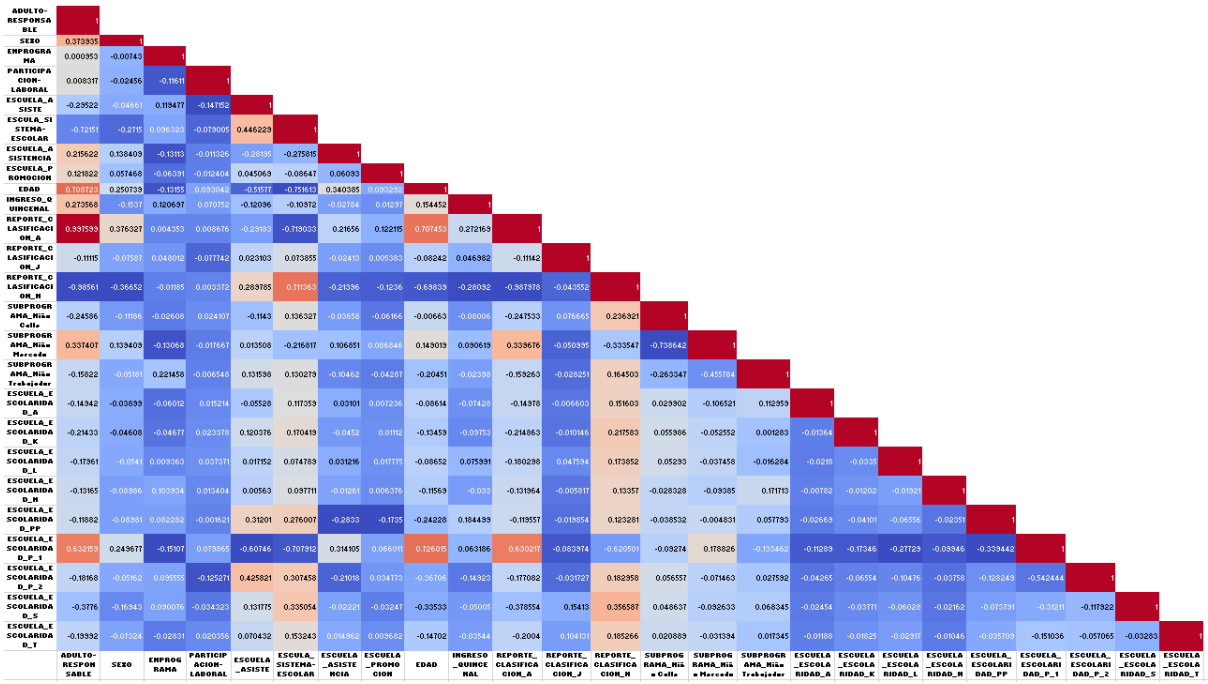

Figura 6.6: Matriz de Correlación en Valores Dummie.

En la Figura 6.6 se identifica la relación que existen entre las variables y que tan fuerte es esa relación. En particular, las variables que denotan la clasificación del reporte (A,J,N) y las relacionadas con la escolaridad resultan tener una correlación aceptable para comenzar a hacer inferencias para el modelo predictivo. Pero, como se busca una comparación entre los Subprogramas, se opta por segmentarlos para generar a cada uno su modelo predictivo.

Para encontrar el mejor modelo se utilizan a los individuos del Subprograma niño mercado, esto al ser el que mayor número de observaciones contiene. Así entonces se procede con el modelado para los demás Subprogramas.

6.1.4. Subprograma Niño Mercado

Se busca conocer como influye cada Subprograma vs cada una de las demás métricas. Por ende, se implementan los modelos para tres bases segmentadas por el Subprograma correspondiente.

[16]:
```
df_NM =
df.drop(['SUBPROGRAMA_Niño Calle',
'SUBPROGRAMA_Niño Trabajador',
'REPORTE_DATE','ID_CUF','ID_NRU'], axis=1)
df_NM_u = df_NM.pop('SUBPROGRAMA_Niño Mercado')
df_NM.insert(22, 'SUBPROGRAMA_Niño Mercado', df_NM_u)
```

Visualización

[17]:
```
X_col = df_NM.columns[:-1]
y_col = df_NM.columns[-1:]
fig, ax = plt.subplots(nrows= 11, ncols=2, figsize=(20,25))
ax_index = []
color = sns.color_palette("flare",11)
for i in range(11):
```

```
        for j in range(2):
                ax_index.append(ax[i,j])
for k in range(len(X_col)):
        ax_index[k].scatter(x = df_NM[X_col[k]], y = df_NM[y_col])
        ax_index[k].set_title(f'{X_col[k]}')
fig.tight_layout()
```

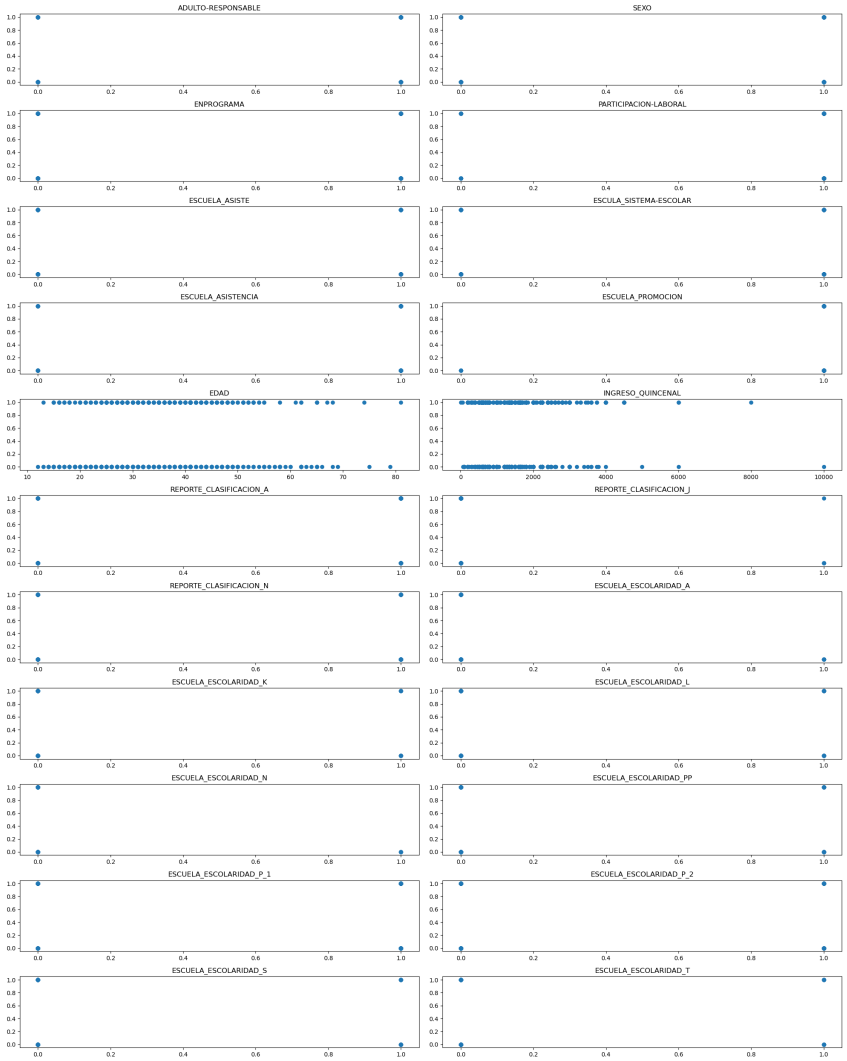

Figura 6.7: Matriz de gráficos de dispersión.

La Figura 6.7 muestra como se comportan los individuos en relación con las demás variables del modelo. Es de notar que casi todas se distribuyen de manera dicotómica exceptuando la edad y el sueldo quincenal. De aquí que se opta por utilizar implementaciones con métodos de clasificación.

6.1.5. Implementación de los Modelos

Se busca encontrar el mejor modelo de clasificación utilizando:

- Regresión Logística.
- K-Vecino más Cercano.
- Árboles de Decisión.

EVALUACIÓN DE MODELOS

Se seleccionan las variables dependientes e independientes para el modelo a implementar y se hace uso de las funciones para la matriz de confusión y la curva de ROC con los argumentos de:

- X: Variables Independientes.
- y: Variable Dependiente.
- model: Modelo de aprendizaje automático para implemetar.
- TestSize: Tamaño del conjunto de prueba dado en número decimal menor a uno y mayor a cero.

Regresión Logística

Con la base de datos ya preparada para su uso, en primer lugar, se implementa un modelo de regresión logística con un tamaño de test del 25 % evaluándolo mediante funciones creadas para graficar la matriz de confusión asociada, al igual que la curva de ROC.

[18]:
```
X = df_NM.drop('SUBPROGRAMA_Niño Mercado', axis =1)
y = df_NM[['SUBPROGRAMA_Niño Mercado']]
y = y['SUBPROGRAMA_Niño Mercado'].tolist()
```

Matriz de Confusión

[19]:
```
from sklearn.linear_model import LogisticRegression as LGR
from funciones import ConfusionMatrixPlot

model = LGR(random_state = 0)
testsize = 0.25

ConfusionMatrixPlot(X,y,model,testsize)
```

```
Aciertos: 170. Fracasos: 86. Precisión: 66.4%
```

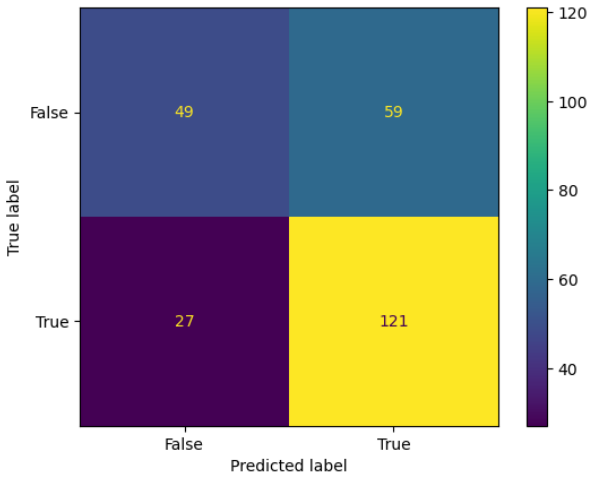

Figura 6.8: Matriz de confusión con regresión logística.

Tomando un cuarto de la base de datos para el entrenamiento, se logra la matriz de confusión de la Figura 6.8 la cual muestra una precisión del 66.4 %. Pero, aunque se tenga una cantidad de aciertos (170) buena, al comparar los falsos positivos contra los falsos negativos se nota una diferencia de 10. La cual, aunque no parezca mucho, denota una mala clasificación del modelo. Generando la curva de ROC correspondiente se tiene:

[20]:
```
from funciones import ROCPlot

ROCPlot(X,y,model,testsize)
```

```
Sin entrenar: ROC AUC=0.500
Entrenado: ROC AUC=0.652
```

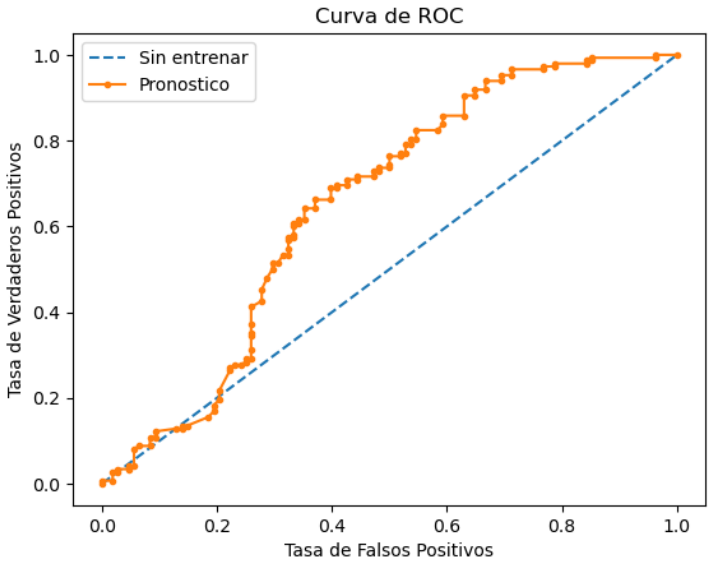

Figura 6.9: Curva de ROC con regresión logística.

La curva de ROC en la Figura 6.9 muestra un decaimiento en la tasa de verdaderos positivos al inicio de la clasificación. Dicho decaimiento es lo que muestra la matriz de confusión con los 10 valores mal clasificados. Una manera de mejorar el modelo es utilizando un tamaño de prueba mayor; pero esto podría llevar a resultados incongruentes. Por lo que se opta por utilizar la eliminación hacia atrás y la selección hacia adelante como sigue:

Eliminación Hacia Atrás

[21]:
```
from mlxtend.feature_selection import SequentialFeatureSelector as fs
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
feature_names=tuple(X.columns)

EhA = fs(
LGR(max_iter=100),
k_features='best',
forward=False,
floating=False,
verbose=2,
scoring='accuracy',
cv=0)
```

```
EhA = EhA.fit(X, y,feature_names);

fig1 = plot_sfs(EhA.get_metric_dict(confidence_interval=0.95), kind='std_err')
plt.title('Puntaje de Regresiones con Eliminación Hacía Atrás')
plt.grid()
plt.show()
```

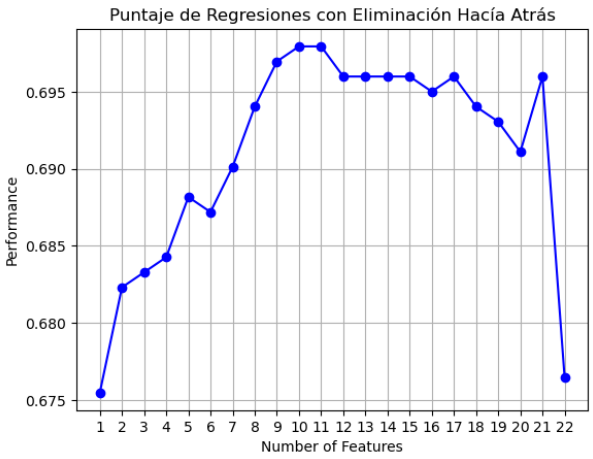

Figura 6.10: Gráfica de Eliminación Hacía Atrás que muestra el score vs. número de variables.

En la gráfica mostrada en la Figura 6.10 se notan los puntajes relacionados con el número de características que se utilizaron para generar el modelo. Es de notar que existe un empate con 10 y 11 características con un puntaje de 0.6979472140762464 en ambos casos. Pero, se toma el de 10 características al ser el más parsimonioso. Dando entonces las características significativas siguientes:

- ENPROGRAMA.
- ESCUELA_ASISTE.
- ESCUELA_SISTEMA-ESCOLAR.
- ESCUELA_ASISTENCIA.
- EDAD.
- REPORTE_CLASIFICACION_J.
- REPORTE_CLASIFICACION_N.
- ESCUELA_ESCOLARIDAD_K.
- ESCUELA_ESCOLARIDAD_L.
- ESCUELA_ESCOLARIDAD_P_2.
- ESCUELA_ESCOLARIDAD_T.

Pero, al no contener la característica de Sueldo Quincenal, se agrega para la implementación. Generando entonces la matriz de confusión y la curva de ROC:

[22]:
```
ncols=list(EhA.k_feature_names_)
ncols.append('INGRESO_QUINCENAL')
```

```
X_EhA_RG= X[ncols]
ConfusionMatrixPlot(X_EhA_RG,y,model,testsize)
```

```
Aciertos: 169. Fracasos: 87. Precisión: 66.015625%
```

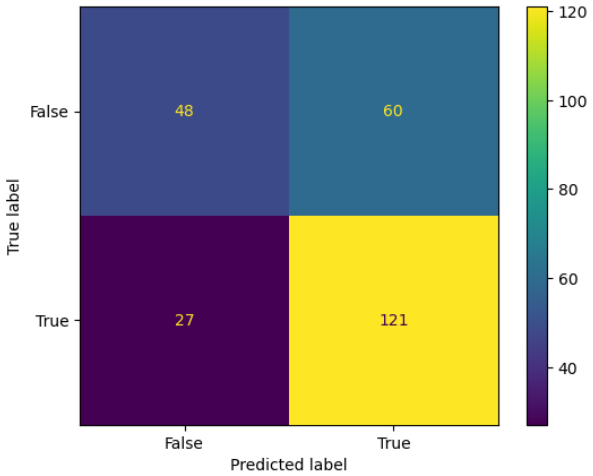

Figura 6.11: Matriz de confusión con Eliminación Hacía Atrás.

[23]:
```
ROCPlot(X_EhA_RG,y,model,testsize)
```

```
Sin entrenar: ROC AUC=0.500
Entrenado: ROC AUC=0.661
```

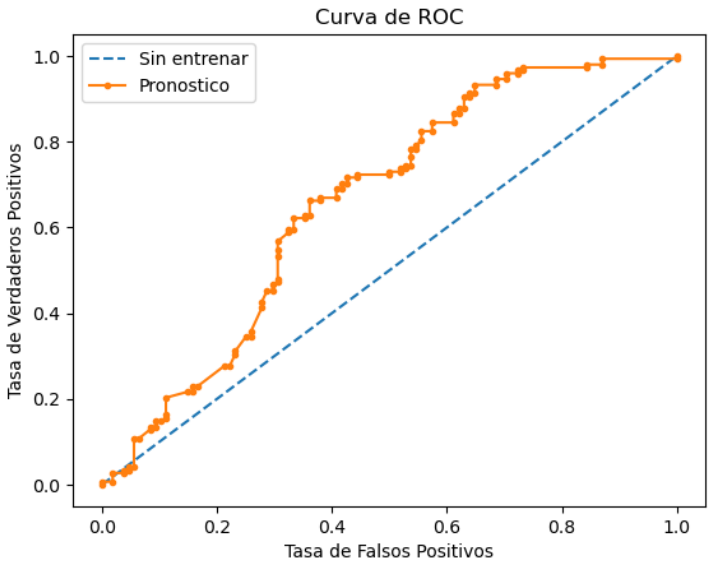

Figura 6.12: Curva de ROC con Eliminación Hacía Atrás.

Utilizando la eliminación hacía atrás se puede notar que el decaimiento del principio se reduce y se desplaza hacía el origen. Esto indica una mejor implementación, además de que el área bajo la curva asciende a 0.661.

Utilizando ahora la selección hacía adelante se tiene lo siguiente:

Selección hacia adelante

[24]:
```
ShA = fs(
LGR(max_iter=100),
k_features='best',
forward=True,
floating=False,
verbose=2,
scoring='accuracy
cv=0)
ShA = ShA.fit(X, y,feature_names);

fig1 = plot_sfs(ShA.get_metric_dict(confidence_interval=0.95), kind='std_err')
plt.title('Puntaje de Regresiones con Selección Hacía Adelate')
plt.grid()
plt.show()
```

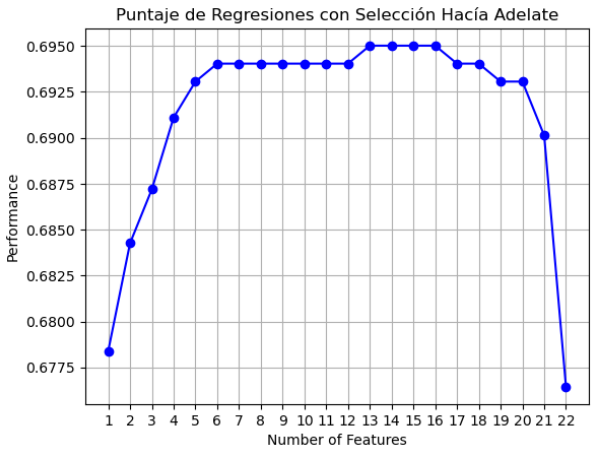

Figura 6.13: Gráfica de Selección Hacía Adelante que muestra el score vs. número de variables

En la Figura 6.13 se muestran los puntajes relacionados con el número de características que se utilizaron para generar el modelo. Existe un empate con 13, 14, 15 y 16 características con un puntaje de 0.6950146627565983 en los casos. Pero, se toma el de 13 características al ser el más parsimonioso. Dando entonces las características de:

- ADULTO-RESPONSABLE.
- SEXO.
- PARTICIPACION-LABORAL.
- ESCUELA_SISTEMA-ESCOLAR.
- ESCUELA_ASISTENCIA.
- ESCUELA_PROMOCION.
- EDAD.

- INGRESO_QUINCENAL.
- REPORTE_CLASIFICACION_A.
- REPORTE_CLASIFICACION_J.
- ESCUELA_ESCOLARIDAD_K.
- ESCUELA_ESCOLARIDAD_N.
- ESCUELA_ESCOLARIDAD_P_1.

Generando entonces la matriz de confusión y la curva de ROC:

[25]:
```
ncols = list(ShA.k_feature_names_)
X_ShA_RG = X[ncols]

ConfusionMatrixPlot(X_ShA_RG,y,model,testsize)
```

```
Aciertos: 171. Fracasos: 85. Precisión: 66.79%
```

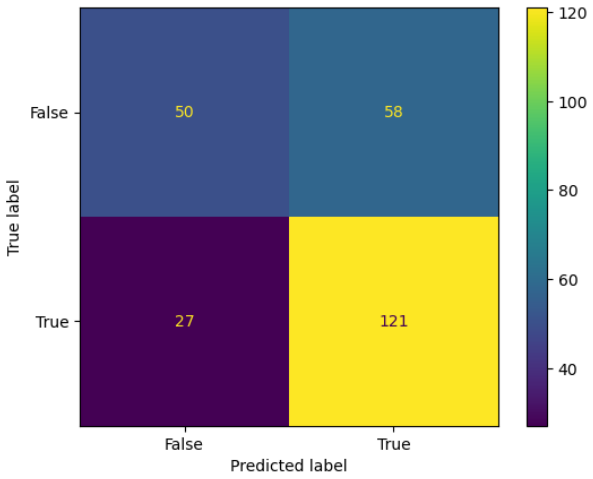

Figura 6.14: Matriz de confusión con Selección Hacía Adelante.

La matriz de confusión de la Figura 6.14 denota una mejora en predecir falsos positivos, pero sigue siendo una implementación insatisfactoria por la diferencia de 8 puntos en el pronóstico resultante de falsos negativos.

Generando la curva de ROC se tiene lo siguiente:

[26]:
```
ROCPlot(X_ShA_RG,y,model,testsize)
```

```
Sin entrenar: ROC AUC=0.500
Entrenado: ROC AUC=0.630
```

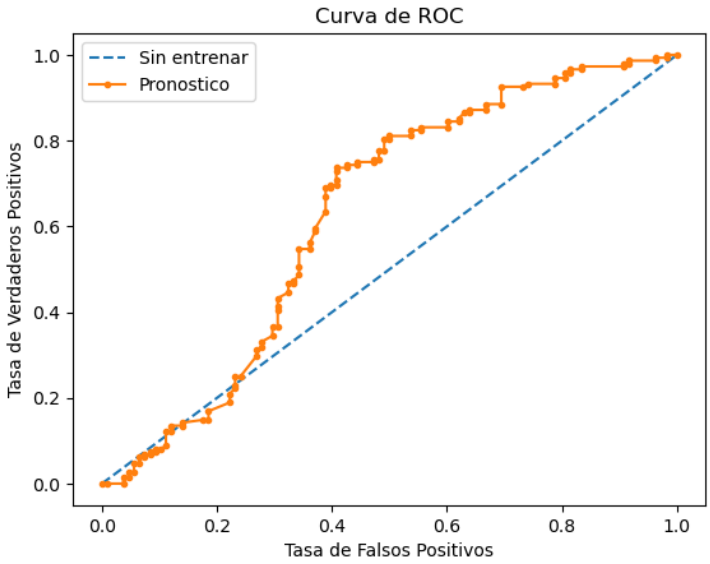

Figura 6.15: Curva de ROC con Selección Hacía Adelante.

El decaimiento de la curva de la Figura 6.15 se extiende sobre la recta identidad denotando una implementación deficiente; además de que la AUC es igual a 0.63 siendo la menor de las tres implementaciones.

Con esto se concluye que, para la implementación con regresión logística, el mejor modelo es el de la eliminación hacía atrás.

6.2. k-Vecino más cercano.

Para conocer el número óptimo de vecinos se tiene que realizar una gráfica comparando distintos valores de vecinos con el puntaje que obtienen al ser implementados. Para continuar con la metodología de selección hacía adelante y eliminación hacía atrás para la búsqueda del mejor modelo con k-NN.

[27]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

testsize = 1/3
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=testsize,random_state=42, stratify=y)
test_p = []; train_p = []; ks=[]

for k in range(1,15):
        knn = KNeighborsClassifier(k)
        knn.fit(X_train,y_train)
        train_p.append(knn.score(X_train,y_train))
        test_p.append(knn.score(X_test,y_test))
        ks.append(k)
```

```
max_train_p = max(train_p)
train_p_ind = [i for i, v in enumerate(train_p) if v == max_train_p]
print('Puntuación máxima de entrenamiento {}
 con k = {}'.format(max_train_p,list(map(lambda x: x+1, train_p_ind))))

max_test_p = max(test_p)
test_p_ind = [i for i, v in enumerate(test_p) if v == max_test_p]
print('Puntuación máxima de prueba {}
 con k = {}'.format(max_test_p,list(map(lambda x: x+1, test_p_ind))))

raw=pd.DataFrame([ks,test_p,train_p]).T
raw.columns=["ks","test_p","train_p"]

p=sns.lineplot(data=raw['test_p'])
p=sns.lineplot(data=raw['train_p'])
```

```
Puntuación máxima de entrenamiento 0.9252199413489736 con k = [1]
Puntuación máxima de prueba 0.6334310850439883 con k = [5]
```

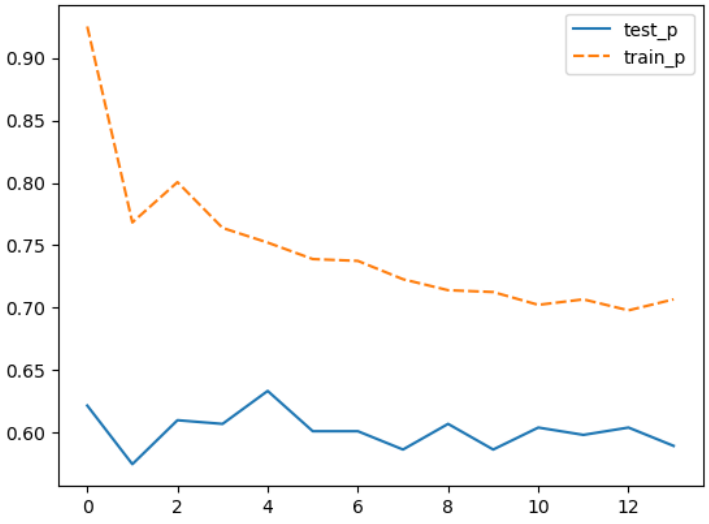

Figura 6.16: Comparación de puntajes con el conjuto de prueba y de entrenamiento.

La Figura 6.16 muestra la comparación entre los puntajes de entrenamiento (línea punteada naranja) y de prueba (línea solida azul). Esto ayuda a observar que el mejor número de vecinos para la implementación es de 5 dando un puntaje de 0.6334310850439883. De modo que, la implementación se hace con los 5 vecinos más cercanos ($k = 5$).

[28]:
```
model = KNeighborsClassifier(5)
ConfusionMatrixPlot(X,y,model,testsize)
```

```
Aciertos: 219. Fracasos: 122. Precisión: 64.22287390029325%
```

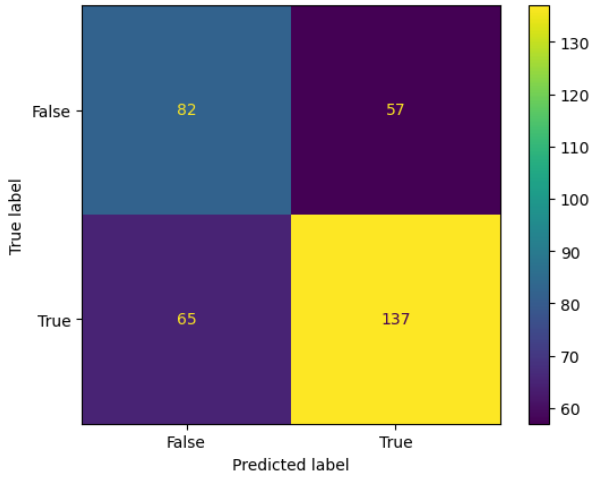

Figura 6.17: Matriz de confusión con todos los valores para k-NN.

La matriz de confusión de la Figura 6.17 denota una buena implementación a la hora de predecir falsos positivos y verdaderos positivos. Teniendo una precisión del 64.22 %.

Generando la curva de ROC se tiene lo siguiente:

[29]:
```
ROCPlot(X,y,model,testsize)
```

```
Sin entrenar: ROC AUC=0.500
Entrenado: ROC AUC=0.692
```

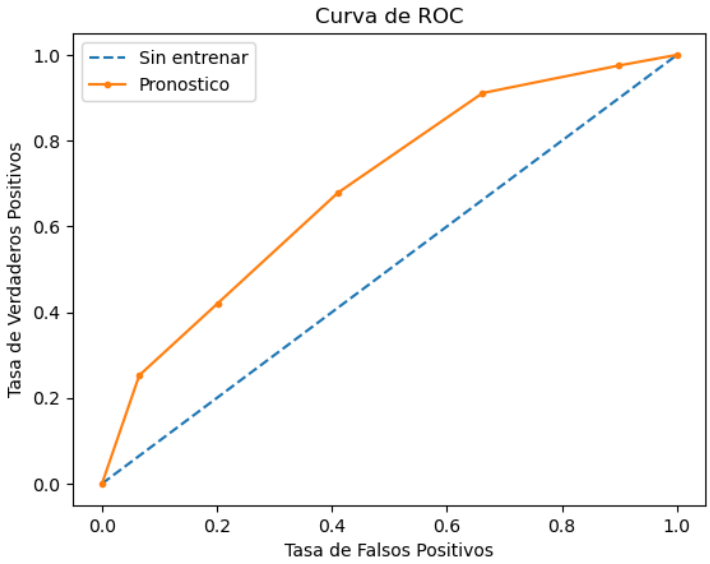

Figura 6.18: Curva de ROC con todos los valores para k-NN.

La curva de ROC de la Figura 6.18 denota una implementación satisfactoria con una AUC de 0.692 dando así un buen acercamiento para generar pronósticos. Pero, para encontrar el mejor modelo, es necesario utilizar la selección hacía adelante y la eliminación hacía atrás.

Selección hacia Adelante

[30]:
```
from mlxtend.feature_selection import SequentialFeatureSelector as fs
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
from sklearn.neighbors import KNeighborsClassifier as knn
feature_names=tuple(X.columns)
ShA_1 = fs(knn(n_neighbors=5),
k_features='best',
forward=True,
floating=False,
verbose=2,
scoring='accuracy',
cv=0)
ShA_1 = ShA_1.fit(X, y,feature_names);

fig1 =
plot_sfs(ShA_1.get_metric_dict(confidence_interval=0.95), kind='std_err')
plt.title('Puntaje de Regresiones con Selección Hacía Adelate')
plt.grid()
plt.show()
```

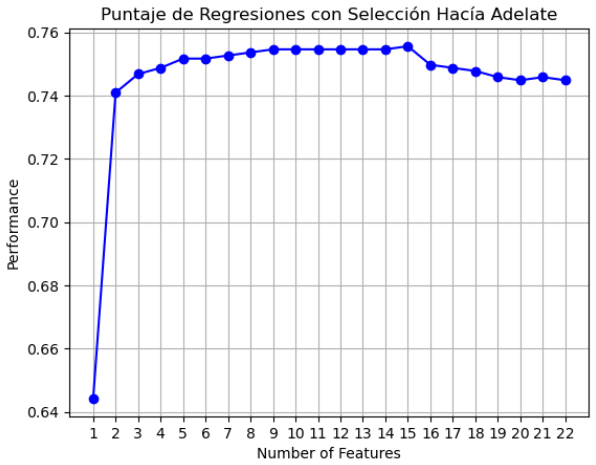

Figura 6.19: Gráfica de Selección Hacía Adelante que muestra el score vs. número de variables

En la gráfica se muestran los puntajes relacionados con el número de características que se utilizaron para generar el modelo. En este caso, el utilizar 15 de las características son las suficientes para generar un modelo predictivo confiable. Dichas características son:

- ENPROGRAMA.
- ESCUELA_ASISTE.
- ESCUELA_SISTEMA-ESCOLAR.
- ESCUELA_ASISTENCIA.
- ESCUELA_PROMOCION.
- EDAD.
- INGRESO_QUINCENAL.

- REPORTE_CLASIFICACION_J.
- ESCUELA_ESCOLARIDAD_A.
- ESCUELA_ESCOLARIDAD_L.
- ESCUELA_ESCOLARIDAD_N.
- ESCUELA_ESCOLARIDAD_PP.
- ESCUELA_ESCOLARIDAD_P_2.
- ESCUELA_ESCOLARIDAD_S.
- ESCUELA_ESCOLARIDAD_T.

Generando entonces la matriz de confusión y la curva de ROC:

[31]:
```
from sklearn.neighbors import KNeighborsClassifier as knn

NewCols = list(ShA_1.k_feature_names_)
model = knn(n_neighbors = 5)
X_SelAdel_knn = X[NewCols]
testsize = 1/3
ConfusionMatrixPlot(X_SelAdel_knn, y, model, testsize)
```

```
Aciertos: 211. Fracasos: 130. Precisión: 61.87683284457478%
```

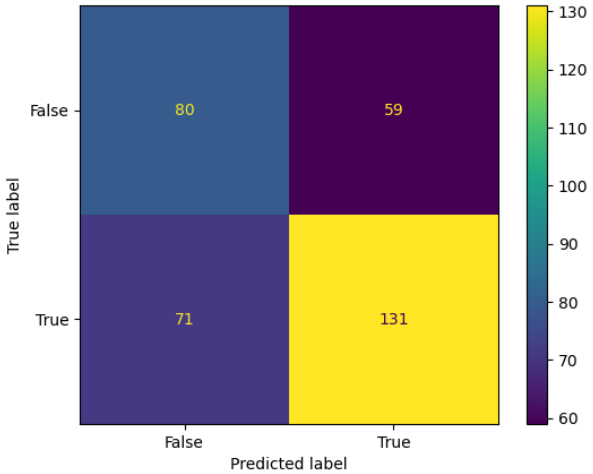

Figura 6.20: Matriz de Confusión Selección Hacia Adelante con Importancia Relativa.

La matriz de confusión denota una buena implementación a la hora de predecir. Teniendo una precisión del 61.87 %, se queda atrás con la implementación pasada. Pero, para tener una mejor comparación se genera la curva de ROC como sigue:

[32]:
```
ROCPlot(X_SelAdel_knn,y,model,testsize)
```

```
Sin entrenar: ROC AUC=0.500
Entrenado: ROC AUC=0.660
```

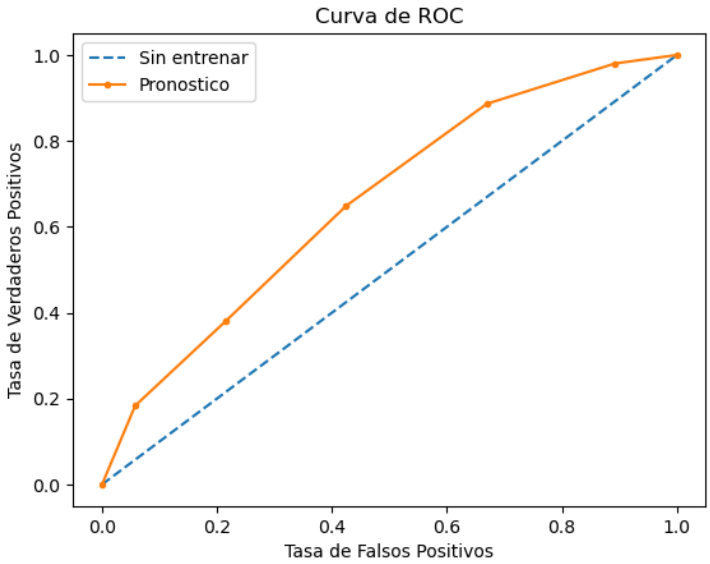

Figura 6.21: Curva de ROC con Importancia Relativa y Selección Hacía Adelante.

La curva de ROC generada denota una implementación satisfactoria con una AUC de 0.66 dando así un buen acercamiento para generar pronósticos, pero se sigue quedando atrás con la implementación anterior. De modo que se continúa con la eliminación hacia atrás:

Eliminación hacia Atrás

[33]:
```
from sklearn.neighbors import KNeighborsClassifier as knn

EhA_knn = fs(knn(n_neighbors=5),
k_features='best',
forward=False,
floating=False,
verbose=2,
scoring='accuracy',
cv=0)
EhA_knn = EhA_knn.fit(X, y,feature_names);

fig1 =
plot_sfs(EhA_knn.get_metric_dict(confidence_interval=0.95), kind='std_err')
plt.title('Puntaje de Regresiones con Selección Hacía Adelate')
plt.grid()
plt.show()
```

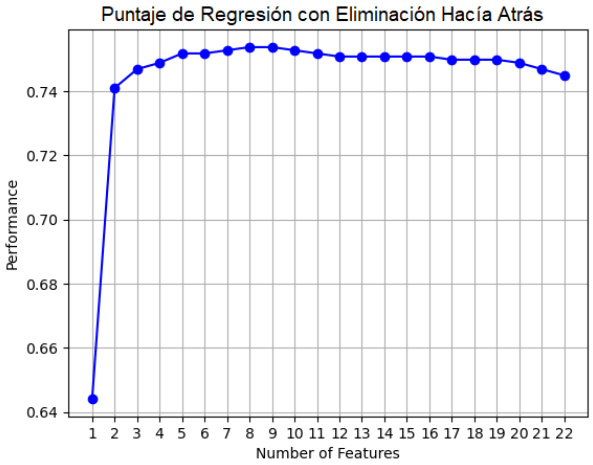

Figura 6.22: Gráfica de puntajes con Selección Hacía Adelante.

En la gráfica de la Figura 6.22 se muestran los puntajes relacionados con el número de características que se utilizaron para generar el modelo. En este caso, el utilizar 9 de las características son las suficientes para generar un modelo predictivo confiable. Dichas características son:

- ENPROGRAMA.
- ESCUELA_ASISTE.
- ESCUELA_ASISTENCIA.
- EDAD.
- INGRESO_QUINCENAL.
- REPORTE_CLASIFICACION_N.
- ESCUELA_ESCOLARIDAD_PP.
- ESCUELA_ESCOLARIDAD_P_2.
- ESCUELA_ESCOLARIDAD_T.

Generando entonces la matriz de confusión y la curva de ROC:

[34]:
```
from sklearn.neighbors import KNeighborsClassifier as knn

from sklearn.neighbors import KNeighborsClassifier as knn

NewCols = list(EhA_knn.k_feature_names_)
X_EhA_knn = X[NewCols]
testsize = 1/3
model = knn(n_neighbors = 5)
ConfusionMatrixPlot(X_EhA_knn, y, model, testsize)
```

```
Aciertos: 210. Fracasos: 131. Precisión: 61.58%
```

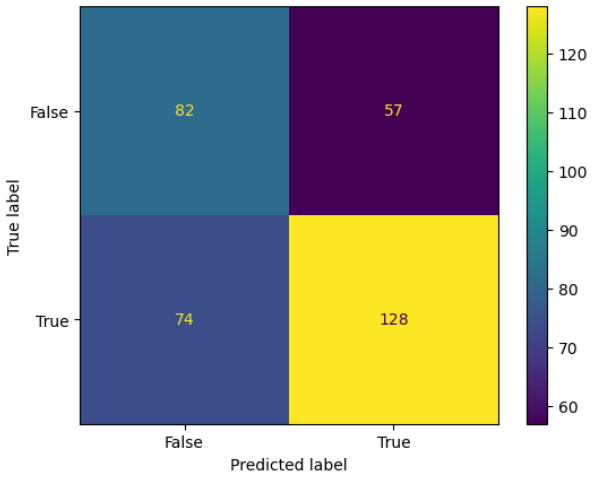

Figura 6.23: Matriz de Confusión Eliminación Hacía Atrás con Importancia Relativa

Nuevamente, la matriz de confusión denota una buena implementación, con una precisión del 61.58 %, se queda aún más atrás con las implementaciones pasadas. Se genera la curva de ROC como sigue:

[35]:
```
ROCPlot(X_EhA_knn,y,model,testsize)
```

```
Sin entrenar: ROC AUC=0.500
Entrenado: ROC AUC=0.687
```

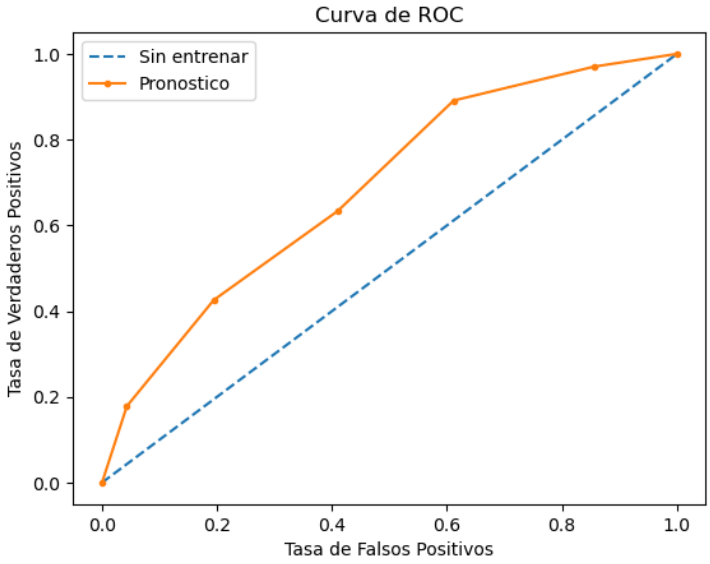

Figura 6.24: Curva de ROC con Importancia Relativa y Eliminación Hacía Atrás.

La implementación, aunque satisfactoria, da una AUC de 0.687. Dando entonces la conclusión de que la implementación utilizando todas las variables es la que mejor se ajusta al modelo.

6.3. Árboles de Decisión

Se busca implementar el modelo de árboles de decisión utilizando el índice de Gini y la Entropía para el reconocimiento del mejor modelo de ajuste y predicción. Esto para ser comparados entre sí y, a su vez, ser evaluados de manera general con los métodos anteriores.

índice de Gini

El índice de Gini se expresa como un número entre cero y uno que denota lo "separado" que se encuentran las instancias entre sí. Un índice de Gini de valor cero representa una segmentación perfecta donde sólo habitan características de la misma clase; mientras que un valor igual a uno representa una segmentación con deficiencias al contener varias instancias diferentes entre sí.

[36]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

depth = 3
testsize = 1/3
X_e,X_p,y_e,y_p=train_test_split(X,y,test_size=testsize,stratify=y)


cad_gini=DecisionTreeClassifier(criterion='gini',
max_depth=depth,
random_state=0)
cad_gini.fit(X_e,y_e)
ypred_gini=cad_gini.predict(X_p)

print('Precisión Utilizando índice de Gini:
{0:0.4f}'.format(accuracy_score(y_p, ypred_gini)))
print('Puntuación Conjunto de Entrenamiento:
{:.4f}'.format(cad_gini.score(X_e,y_e)))
print('Puntuación Conjunto de Prueba:
{:.4f}'.format(cad_gini.score(X_p, y_p)))
print('Diferencia Absoluta de la Puntuación de los Conjuntos:
{0:0.4f}'.format(abs(cad_gini.score(X_e, y_e) - cad_gini.score(X_p, y_p))))

plt.subplots(figsize = (10,10))
tree.plot_tree(cad_gini.fit(X_e, y_e));
fig.tight_layout();
```

```
Precisión Utilizando índice de Gini:0.6891
Puntuación Conjunto de Entrenamiento:0.7126
Puntuación Conjunto de Prueba: 0.6891
Diferencia Absoluta de la Puntuación de los Conjuntos:0.0235
```

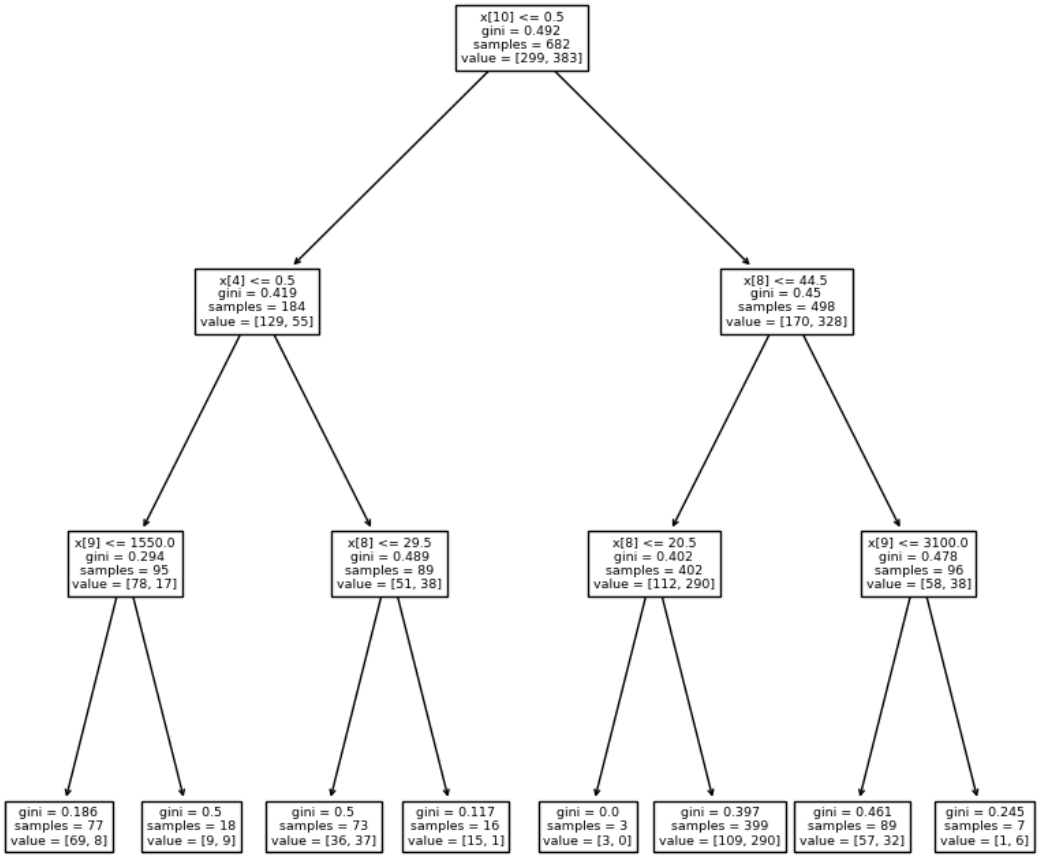

Figura 6.25: Árbol de decisión utilizando índice de Gini.

Como se muestra en la Figura 6.25, utilizando todas la características para el modelo, con una profundidad de tres se logran 8 segmentaciones, donde una de ellas es de índice de Gini cero. Es decir, representa una segmentación perfecta para las características. En este caso en particular, el mayor valor alcanzado en el índice de Gini es de 0.5.

Calculando entonces la matriz de correlación y la curva de ROC se tiene lo siguiente:

[37]:
```
ConfusionMatrixPlot(X,y,cad_gini,testsize)
```

```
Aciertos: 234. Fracasos: 107. Precisión: 68.62%
```

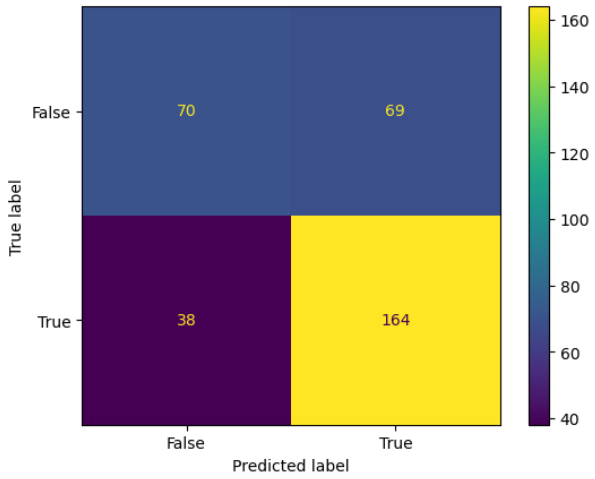

Figura 6.26: Matriz de confusión con índice de Gini.

La matriz de confusión de la Figura 6.26 revela una implementación satisfactoria, en cuanto a aciertos se refiere. Aunque la diferencia de una unidad en los valores de falsos positivos llega a ser de índole, dígase, preocupante para el desarrollo de una predicción; la diferencia de 126 unidades a favor de los verdaderos positivos hace que sea un modelo apto para generar predicciones.

[38]:
```
        ROCPlot(X,y,cad_gini,testsize)
```

```
Sin entrenar: ROC AUC=0.500
Entrenado: ROC AUC=0.726
```

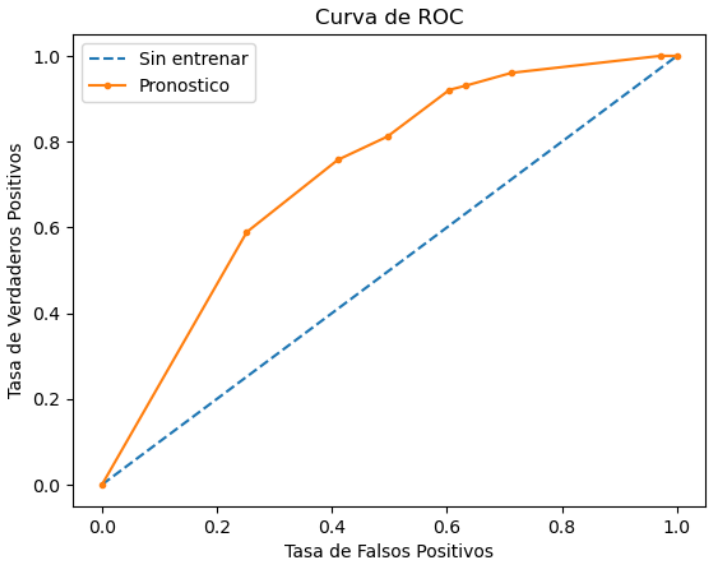

Figura 6.27: Curva de ROC con índice de Gini.

Por otro lado, la curva de ROC de la Figura 6.27 muestra un AUC de 0.726; lo cual indica una relación bastante aceptable entre los valores reales y la implementación del modelo.

Se procede a calcular la importancia relativa de las características; la cual es una técnica que asigna una puntuación a las características de entrada en función de su utilidad para predecir una variable objetivo. Es decir, se puntuan las características que aportan más al modelo para generar una predicción.

[39]:
```
plt.figure(figsize = (8,4))
etiquetas = ['ADULTO-RESPONSABLE','SEXO','ENPROGRAMA','PARTICIPACION-LABORAL',
'ESCUELA_ASISTE','ESCUELA_SISTEMA-ESCOLAR','ESCUELA_ASISTENCIA',
'ESCUELA_PROMOCION', 'EDAD', 'INGRESO_QUINCENAL','REPORTE_CLASIFICACION_A',
'REPORTE_CLASIFICACION_J','REPORTE_CLASIFICACION_N',
'ESCUELA_ESCOLARIDAD_A','ESCUELA_ESCOLARIDAD_K',
'ESCUELA_ESCOLARIDAD_L','ESCUELA_ESCOLARIDAD_N',
'ESCUELA_ESCOLARIDAD_PP','ESCUELA_ESCOLARIDAD_P_1',
'ESCUELA_ESCOLARIDAD_P_2','ESCUELA_ESCOLARIDAD_S', 'ESCUELA_ESCOLARIDAD_T']
importancia = cad_gini.feature_importances_
índices = np.argsort(importancia)
plt.title('Importancia de Características')
plt.barh(range(len(índices)), importancia[índices], color = 'b',
align = 'center')
plt.yticks(range(len(índices)), [etiquetas[i] for i in índices])
plt.xlabel('Importancia Relativa')
plt.show()
```

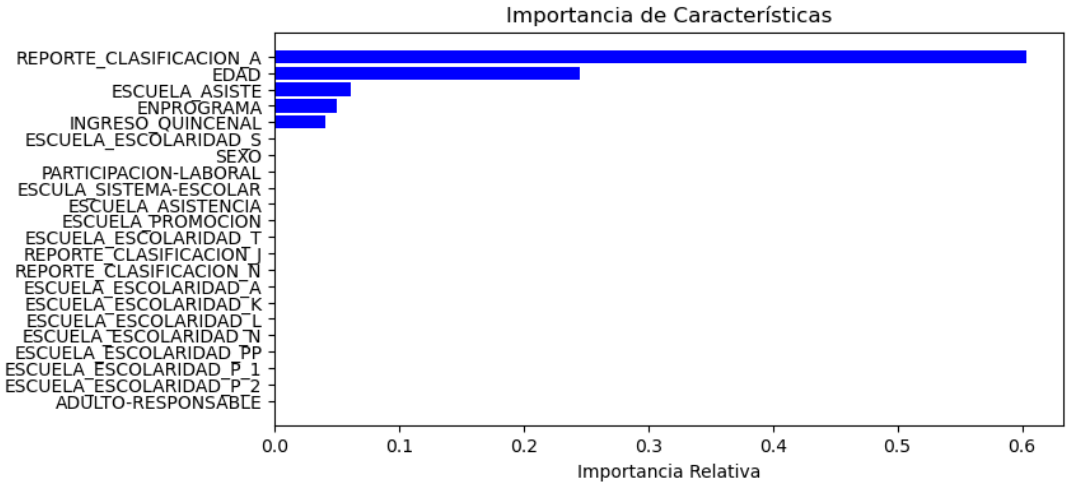

Figura 6.28: Importancia relativa de las características con índice de Gini.

Al hacer el procedimiento, la gráfica de la Figura 6.28 indica que las características más importantes para hacer pronóstico son:

- REPORTE_CLASIFICACION_A.
- EDAD.
- ESCUELA_ASISTE.
- ENPROGRAMA.
- INGRESO_QUINCENAL.

Realizando la implementación utilizando sólo estas características, el modelo queda de la siguiente forma:

Índice de Gini con Importancia Relativa

[40]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split


X_tree_gini_fi=
X[['REPORTE_CLASIFICACION_A','EDAD','ESCUELA_ASISTE','ENPROGRAMA',
'INGRESO_QUINCENAL']]


testsize = 1/3
depth = 3
X_e,X_p,y_e,y_p=
train_test_split(X_tree_gini_fi,y,test_size=testsize,stratify=y)


cad_gini = DecisionTreeClassifier(criterion =
'gini',max_depth=depth,random_state=0)
cad_gini.fit(X_e,y_e)
ypred_gini = cad_gini.predict(X_p)


print('Precisión Utilizando índice de Gini:
{0:0.4f}'.format(accuracy_score(y_p, ypred_gini)))
print('Puntuación Conjunto de Entrenamiento:
{:.4f}'.format(cad_gini.score(X_e,y_e)))
print('Puntuación Conjunto de Prueba:
{:.4f}'.format(cad_gini.score(X_p, y_p)))
print('Diferencia Absoluta de la Puntuación de los Conjuntos:
{0:0.4f}'.format(abs(cad_gini.score(X_e, y_e) - cad_gini.score(X_p, y_p))))


plt.subplots(figsize = (10,10))
tree.plot_tree(cad_gini.fit(X_e, y_e));
fig.tight_layout();
```

```
Precisión Utilizando índice de Gini:0.6833
Puntuación Conjunto de Entrenamiento:0.7097
Puntuación Conjunto de Prueba: 0.6833
Diferencia Absoluta de la Puntuación de los Conjuntos:0.0264
```

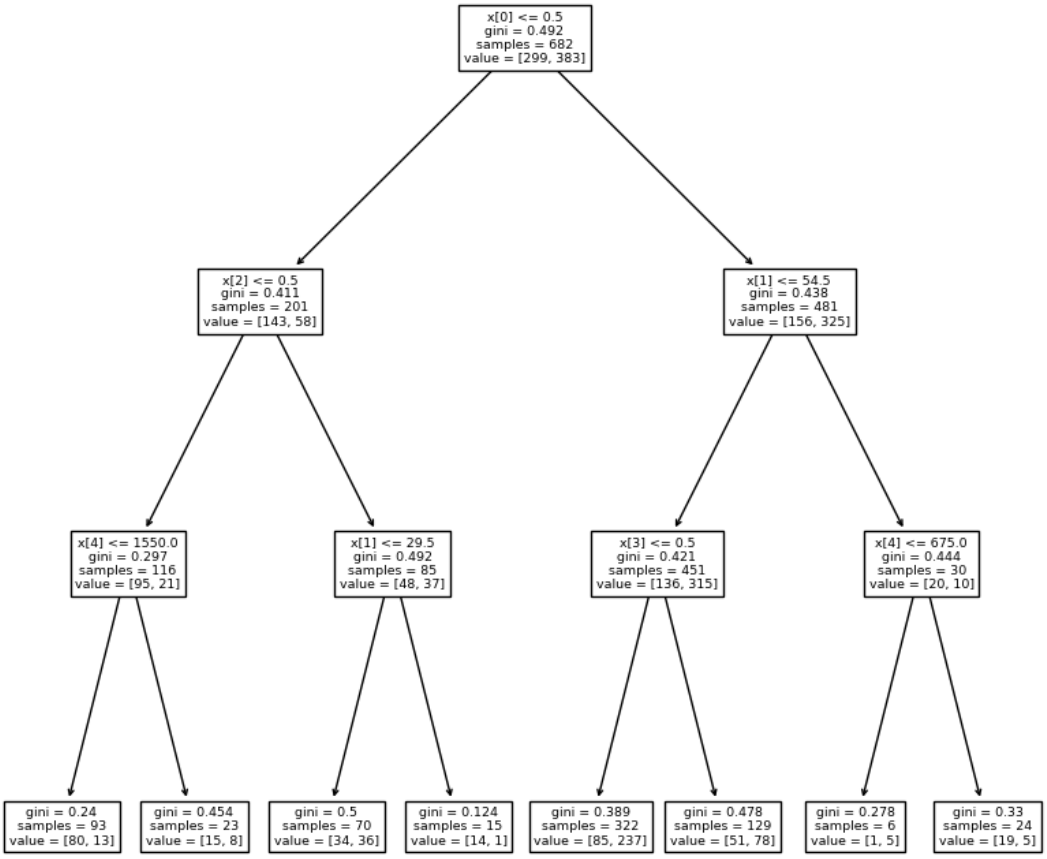

Figura 6.29: Árbol de decisión utilizando índice de Gini con importancia relativa.

En el árbol generado en la Figura 6.29 se puede notar que, en efecto, la implemetación utilizando todas las características y usando las que se calcularon con la importancia relativa, son exactamente la misma. Esto hace que se termine generando el mismo árbol de decisión y, es de esperar, la misma matriz de confusión y la misma curva de ROC:

[41]:
```
ConfusionMatrixPlot(X_tree_gini_fi,y,cad_gini,testsize)
```

```
Aciertos: 234. Fracasos: 107. Precisión: 68.62170087976538%
```

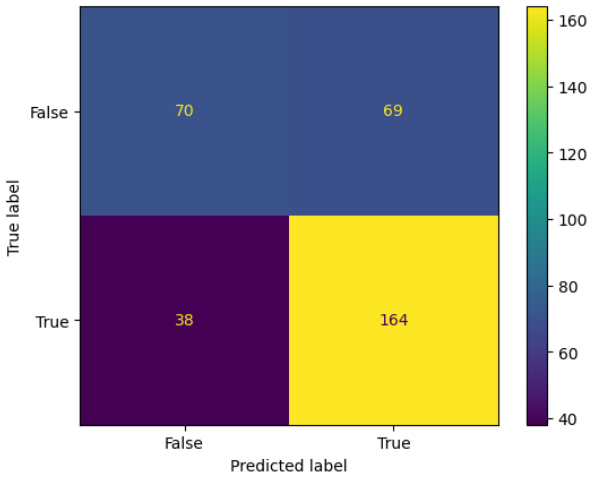

Figura 6.30: Matriz de confusión índice de Gini con importancia relativa.

Como se puede notar, la matriz de confusión de la Figura 6.30 es la misma que de la Figura 6.26

[42]:
```
ROCPlot(X_tree_gini_fi,y,cad_gini,testsize)
```

```
Sin entrenar: ROC AUC=0.500
Entrenado: ROC AUC=0.726
```

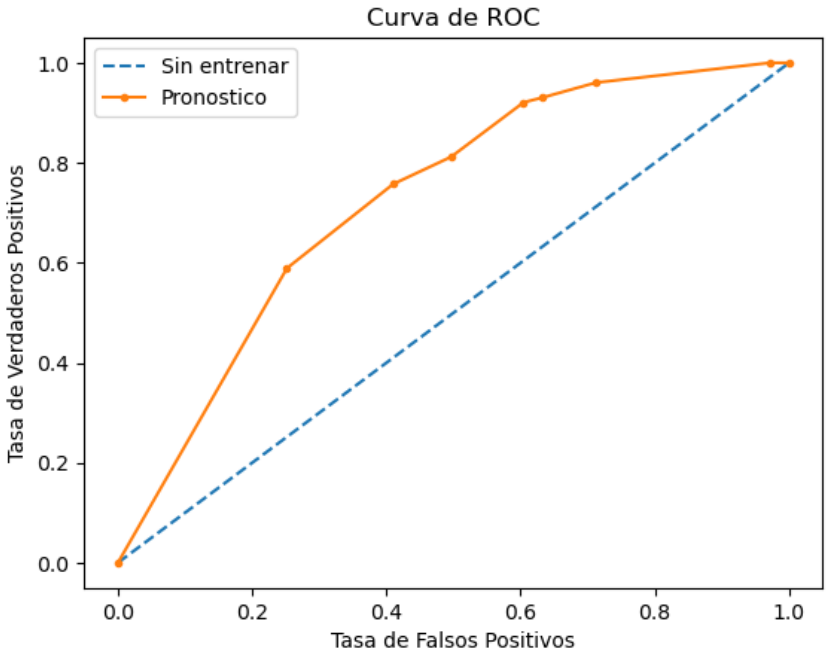

Figura 6.31: Curva de ROC índice de Gini con importancia relativa.

De igual forma, la curva de ROC de la Figura 6.31 es la misma que de la Figura 6.27

Con esto se puede notar que, en efecto, solo las variables de:

- REPORTE_CLASIFICACION_A.
- EDAD.
- ESCUELA_ASISTE.
- ENPROGRAMA.
- INGRESO_QUINCENAL.

son las que influyen directamente en el modelo de árboles de decisión utilizando el índice de Gini.

Utilizando esta misma lógica se genera entonces la implementación con la entropía para generar el modelo.

Entropía

La entropía es una forma de cuantificar el desorden de un sistema. En el caso de los nodos de los árboles, el desorden se entiende como la impureza. Si un nodo es puro, contiene únicamente observaciones de una clase, su entropía es cero. Por el contrario, si el valor es 1, el nodo se considera impuro.

[43]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

depth = 3
testsize = 1/3
X_e,X_p,y_e,y_p=train_test_split(X,y,test_size=testsize,stratify=y)


cad_ent = DecisionTreeClassifier(criterion =
'entropy',max_depth=depth,random_state=0)
cad_ent.fit(X_e,y_e)
ypred_ent = cad_ent.predict(X_p)

print('Precisión Utilizando Entropía:
{0:0.4f}'.format(accuracy_score(y_p, ypred_ent)))

print('Puntuación Conjunto de Entrenamiento:
{:.4f}'.format(cad_ent.score(X_e,y_e)))

print('Puntuación Conjunto de Prueba:
{:.4f}'.format(cad_ent.score(X_p, y_p)))

print('Diferencia Absoluta de la Puntuación de los Conjuntos:
{0:0.4f}'.format(abs(cad_ent.score(X_e, y_e) - cad_ent.score(X_p, y_p))))

plt.subplots(figsize = (10,10))
tree.plot_tree(cad_ent.fit(X_e, y_e));
fig.tight_layout();
```

```
Precisión Utilizando Entropía:0.6452
Puntuación Conjunto de Entrenamiento:0.7185
Puntuación Conjunto de Prueba: 0.6452
Diferencia Absoluta de la Puntuación de los Conjuntos:0.0733
```

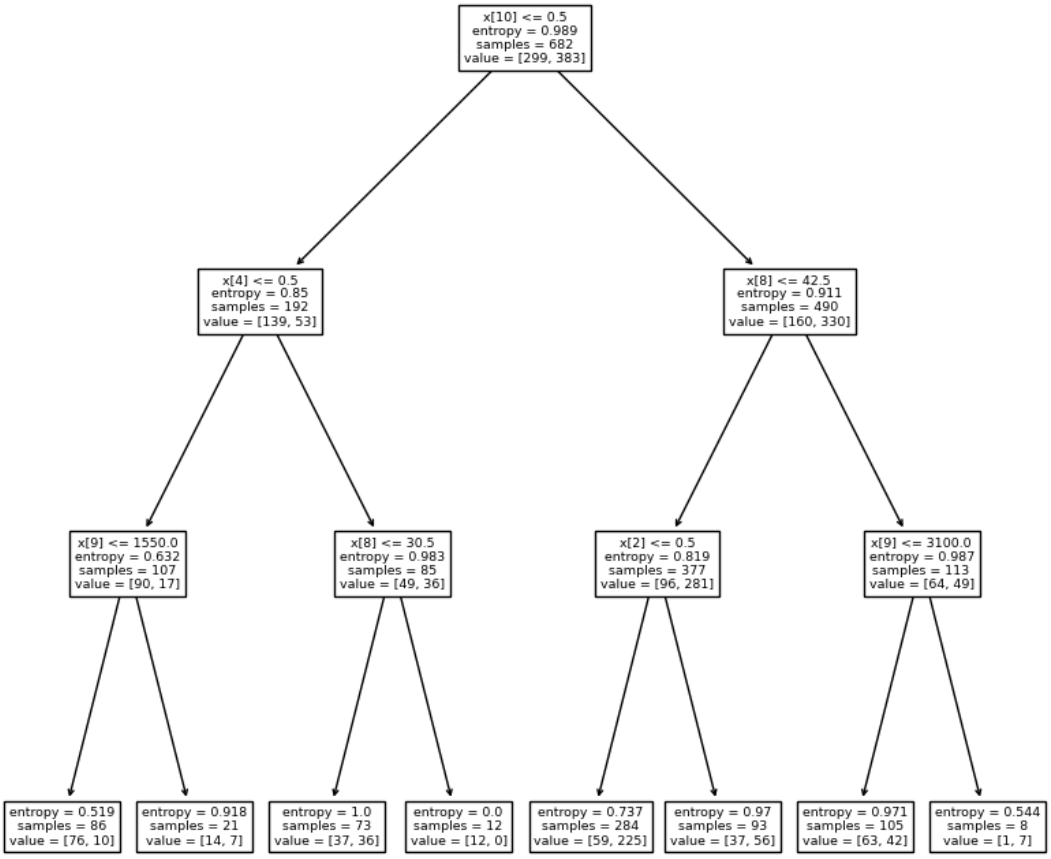

Figura 6.32: Árbol de decisión utilizando Entropía.

El árbol generado de la Figura 6.32 con la entropía y una profundidad de tres genera 8 segmentaciones, donde una se considera pura. En el caso de la entropía se nota que la clasificación no es satisfactoria dados los valores generados al final cercanos, y uno igual, a uno. Es decir, la implementación de árboles de decisión utilizando la entropía no es apta para el modelo. Por lo que queda descartado y solo se toman en cuenta los valores con el índice de Gini con la importancia relativa.

Comparación de Implementaciones

El mejor modelo para generar pronósticos es el de árboles de decisión. Para ejemplificar mejor esto se muestra la siguiente tabla con los valores que se obtuvieron de cada modelo:

En la tabla se muestra que, en efecto, el modelo con árboles de decisión es el más apto para generar el modelo predictivo.

Tomando esto en cuenta, se utiliza para generar las curvas de ROC para los demás Subprogramas.

6.4. Subprograma Niño Calle

[44]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

df_NC = df.drop(['SUBPROGRAMA_Niño Mercado',
'SUBPROGRAMA_Niño Trabajador','REPORTE_DATE','ID_CUF','ID_NRU'], axis=1)
df_NC_u = df_NC.pop('SUBPROGRAMA_Niño Calle')
df_NC.insert(22, 'SUBPROGRAMA_Niño Calle', df_NC_u)

# Selección de variables
X = df_NC.drop('SUBPROGRAMA_Niño Calle', axis =1)
y = df_NC[['SUBPROGRAMA_Niño Calle']]
y = y['SUBPROGRAMA_Niño Calle'].tolist()

X =
X[['REPORTE_CLASIFICACION_A','EDAD',
'ESCUELA_ASISTE','ENPROGRAMA','INGRESO_QUINCENAL']]

testsize = 1/3
depth = 3
X_e,X_p,y_e,y_p=train_test_split(X,y,test_size=testsize,stratify=y)


cad_gini = DecisionTreeClassifier(criterion =
'gini',max_depth=depth,random_state=0)
cad_gini.fit(X_e,y_e)
ypred_gini = cad_gini.predict(X_p)

print('Precisión Utilizando índice de Gini:
{0:0.4f}'.format(accuracy_score(y_p, ypred_gini)))
print('Puntuación Conjunto de Entrenamiento:
{:.4f}'.format(cad_gini.score(X_e,y_e)))
print('Puntuación Conjunto de Prueba:
{:.4f}'.format(cad_gini.score(X_p, y_p)))
print('Diferencia Absoluta de la Puntuación de los Conjuntos:
{0:0.4f}'.format(abs(cad_gini.score(X_e, y_e) - cad_gini.score(X_p, y_p))))

plt.subplots(figsize = (10,10))
tree.plot_tree(cad_gini.fit(X_e, y_e));
fig.tight_layout();
```

```
Precisión Utilizando índice de Gini:0.7390
Puntuación Conjunto de Entrenamiento:0.7566
Puntuación Conjunto de Prueba: 0.7390
Diferencia Absoluta de la Puntuación de los Conjuntos:0.0176
```

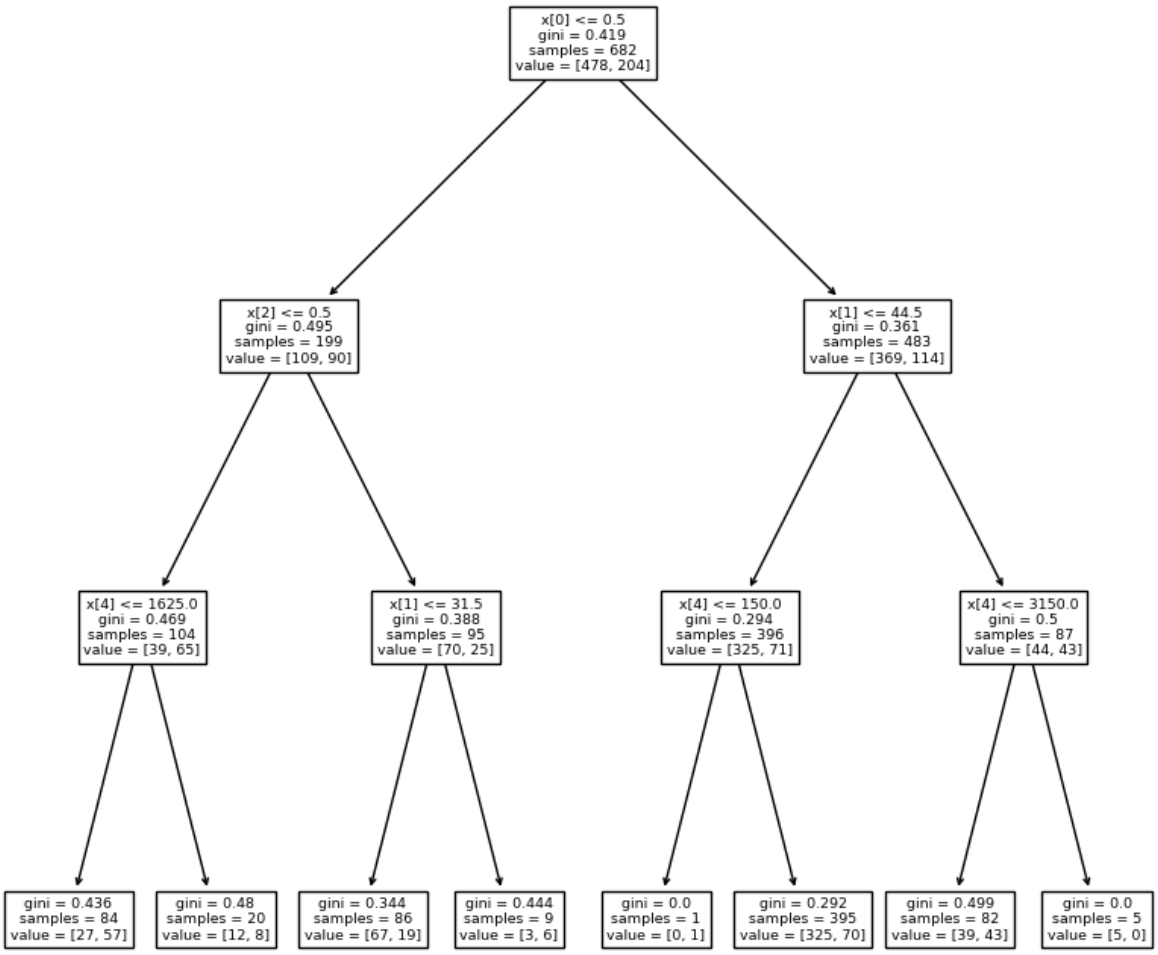

Figura 6.33: Subprograma Niño Calle: Árbol de decisión con índice de Gini e importancia relativa.

[45]:
```
ROCPlot(X,y,cad_gini,testsize)
```

```
Sin entrenar: ROC AUC=0.500
Entrenado: ROC AUC=0.718
```

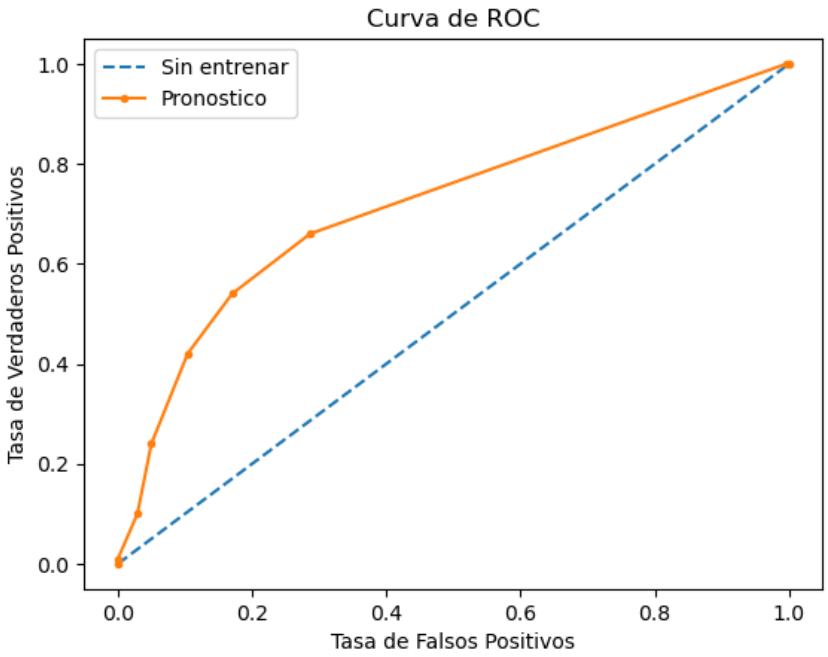

Figura 6.34: Subprograma Niño Calle: Curva de ROC con índice de Gini e importancia relativa.

6.5. Subprograma Niño Trabajador

[46]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split


df_NT = df.drop(['SUBPROGRAMA_Niño Mercado',
'SUBPROGRAMA_Niño Calle','REPORTE_DATE','ID_CUF','ID_NRU'], axis=1)

df_NT_u = df_NT.pop('SUBPROGRAMA_Niño Trabajador')

df_NT.insert(22, 'SUBPROGRAMA_Niño Trabajador', df_NT_u)


# Seleccion de variables
X = df_NT.drop('SUBPROGRAMA_Niño Trabajador', axis =1)
```

```
y = df_NT[['SUBPROGRAMA_Niño Trabajador']]
y = y['SUBPROGRAMA_Niño Trabajador'].tolist()


X =
X[['REPORTE_CLASIFICACION_A','EDAD','ESCUELA_ASISTE',
'ENPROGRAMA','INGRESO_QUINCENAL']]


testsize = 1/3
depth = 3
X_e,X_p,y_e,y_p=train_test_split(X,y,test_size=testsize,stratify=y)


cad_gini = DecisionTreeClassifier(criterion =
'gini',max_depth=depth,random_state=0)
cad_gini.fit(X_e,y_e)
ypred_gini = cad_gini.predict(X_p)


print('Precisión Utilizando índice de Gini:
{0:0.4f}'.format(accuracy_score(y_p, ypred_gini)))


print('Puntuación Conjunto de Entrenamiento:
{:.4f}'.format(cad_gini.score(X_e,y_e)))


print('Puntuación Conjunto de Prueba:
{:.4f}'.format(cad_gini.score(X_p, y_p)))


print('Diferencia Absoluta de la Puntuación de los Conjuntos:
{0:0.4f}'.format(abs(cad_gini.score(X_e, y_e) - cad_gini.score(X_p, y_p))))


plt.subplots(figsize = (10,10))
tree.plot_tree(cad_gini.fit(X_e, y_e));
fig.tight_layout();
```

```
Precisión Utilizando índice de Gini:0.8475
Puntuación Conjunto de Entrenamiento:0.8768
Puntuación Conjunto de Prueba: 0.8475
Diferencia Absoluta de la Puntuación de los Conjuntos:0.0293
```

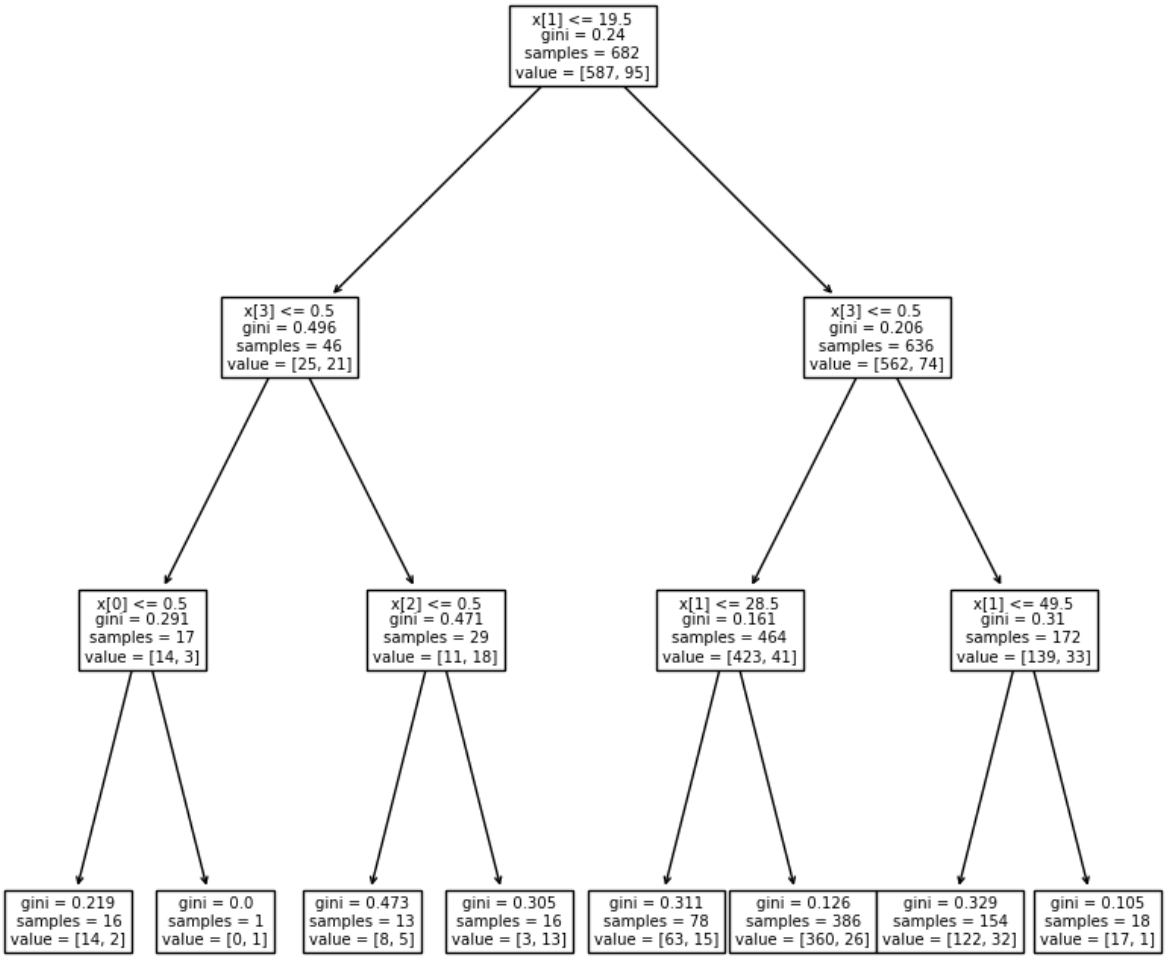

Figura 6.35: Subprograma Niño Trabajador: Árbol de decisión con índice de Gini e importancia relativa.

[47]:
```
ROCPlot(X,y,cad_gini,testsize)
```

```
Sin entrenar: ROC AUC=0.500
Entrenado: ROC AUC=0.625
```

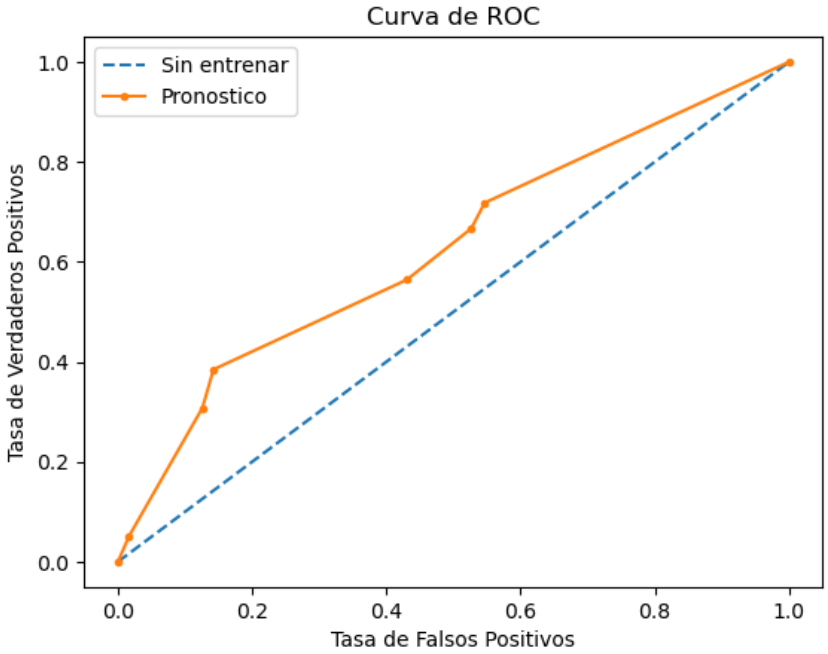

Figura 6.36: Subprograma Niño Trabajador: Curva de ROC con índice de Gini e importancia relativa.

Despliegue

Se genera una comparación del modelo realizado contra los resultados reales. Cada base de datos utilizada tiene por columnas:

- REPORTE_CLASIFICACION_A.
- EDAD.
- ESCUELA_ASISTE.
- ENPROGRAMA.
- INGRESO_QUINCENAL.
- Subprograma Correspondiente: Niño_Mercado, Niño_Trabajador o Niño_Calle.

De las cuales se genera el modelo implementado anteriormente.

6.6. Pronóstico: Niño Calle

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.model_selection import train_test_split

df_R_nc=pd.read_excel('df_Tree_Gini_NC.xlsx', sheet_name='real')

X_tree_gini_fi=
df_R_nc[['REPORTE_CLASIFICACION_A','EDAD',
'ESCUELA_ASISTE',
'ENPROGRAMA','INGRESO_QUINCENAL']]
y = df_R_nc[['Niño Calle']]
y = y['Niño Calle'].tolist()
testsize = 1/3
depth = 3
X_e,X_p,y_e,y_p=
train_test_split(X_tree_gini_fi,y,test_size=testsize,stratify=y)

cad_gini = DecisionTreeClassifier(criterion =
'gini',max_depth=depth,random_state=0)
cad_gini.fit(X_e,y_e)
ypred_gini = cad_gini.predict(X_p)

y_pros_R= pd.DataFrame(cad_gini.predict_proba(X_tree_gini_fi))
y_pros_R
```

[1]:

No	0	1
1	0.838308	0.161692
2	0.346939	0.653061
3	0.838308	0.161692
4	0.559322	0.440678
5	0.559322	0.440678
...	...	...
1019	0.838308	0.161692
1020	0.7375	0.2625
1021	0.838308	0.161692
1022	0.7375	0.2625
1023	0.838308	0.161692

Tabla 6.1: Tabla de probabilidades basado en el modelo para la predicción con árboles de decisión con índice de Gini.

La Tabla 6.1 muestra la probabilidad de que el individuo pertenezca, o no, al Subprograma Niño Calle. Las entradas de la tabla deben ser transformadas a valores de 0 y 1 dependiendo de la probabilidad. Para clasificar las entradas de la tabla se toman como 1 los que tengan probabilidad arriba de $\frac{1}{2}$ y 0 para el resto de valores de la siguiente manera:

[2]:
```
vals_1_r = []
vals_2_r = []

for pros in y_pros_R[0]:
        if pros <= 0.5:
                vals_1_r.append(0)
        else:
```

```
                vals_1_r.append(1)

for pros in y_pros_R[1]:
        if pros <= 0.5:
                vals_2_r.append(0)
        else:
                vals_2_r.append(1)

pronosticos_R = pd.DataFrame([vals_1_r,vals_2_r]).T
pronosticos_R
```

[2]:

No	0	1
1	1	0
2	0	1
3	1	0
4	1	0
5	1	0
...	...	...
1019	1	0
1020	1	0
1021	1	0
1022	1	0
1023	1	0

Tabla 6.2: Tabla de valores transformados a ceros y unos.

La Tabla 6.2 redondea los valores de la Tabla 6.1 para poder gratificarlos en un histograma como sigue:

[3]:
```
labels = ['No Pertenece','Pertenece']
NPertenece_R =len(y)-sum(y)
SPertenece_R = sum(y)
NPertenece_P = sum(pronosticos_R[0])
SPertenece_P = sum(pronosticos_R[1])
vals_P = [NPertenece_P, SPertenece_P]
vals_R = [NPertenece_R, SPertenece_R]
corde = np.arange(len(vals_P))
ancho = 0.3

fig, ax = plt.subplots()
ax.bar(corde-ancho/2, vals_P, ancho, label='Pronóstico')
ax.bar(corde+ancho/2, vals_R, ancho, label ='Real')

for i,j in zip(corde, vals_P):
        ax.annotate(j, xy=(i-0.2, j+0.2))
for i,j in zip(corde, vals_R):
        ax.annotate(j, xy=(i+0.1, j+0.2))

ax.set_title('Gráfico de Pronóstico v.s Real: Niño Calle')
ax.set_xticks(corde)
ax.set_xticklabels(labels)
```

```
plt.legend();
```

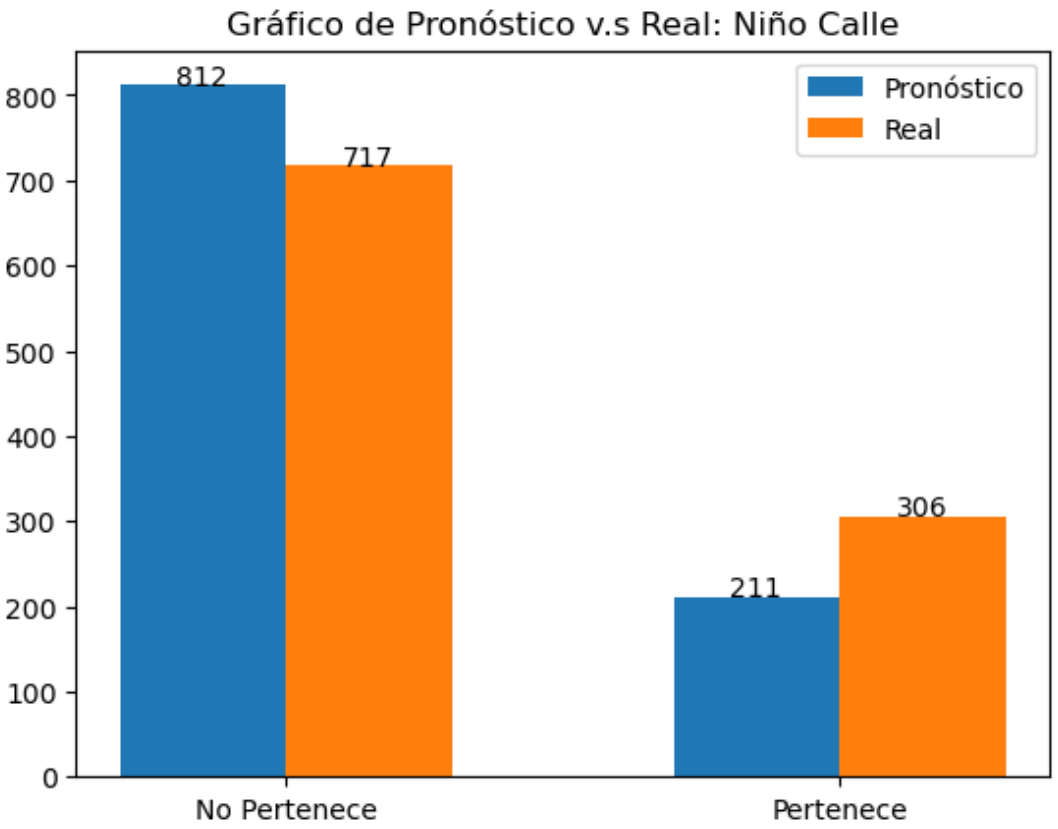

Figura 6.37: Gráfico de barras que muestra en el eje X si el individuo pertenece, o no, al Subprograma Niño Calle. Cada barra muestra el total de individuos dentro de cada categoría segmentados por origen (real o pronóstico).

En la Figura 6.37 se puede notar que, aunque los valores no son los mismos, la aproximación realizada es bastante aceptable. Dando entonces pie a realizar la comparativa con los demás modelos

6.7. Pronóstico: Niño Mercado

[4]:
```
df_R_nm=pd.read_excel('df_Tree_Gini_NM.xlsx', sheet_name='real')

X_tree_gini_fi=df_R_nm[['REPORTE_CLASIFICACION_A','EDAD',
'ESCUELA_ASISTE',
'ENPROGRAMA','INGRESO_QUINCENAL']]
y = df_R_nm[['Niño Mercado']]
y = y['Niño Mercado'].tolist()
testsize = 1/3
depth = 3
X_e,X_p,y_e,y_p=
train_test_split(X_tree_gini_fi,y,test_size=testsize,stratify=y)
```

```
cad_gini = DecisionTreeClassifier(criterion =
'gini',max_depth=depth,random_state=0)
cad_gini.fit(X_e,y_e)
ypred_gini = cad_gini.predict(X_p)

y_pros_R= pd.DataFrame(cad_gini.predict_proba(X_tree_gini_fi))

vals_1_r = []
vals_2_r = []

for pros in y_pros_R[0]:
        if pros<=0.5:
                vals_1_r.append(0)
        else:
                vals_1_r.append(1)

for pros in y_pros_R[1]:
        if pros<=0.5:
                vals_2_r.append(0)
        else:
                vals_2_r.append(1)

pronosticos_R = pd.DataFrame([vals_1_r,vals_2_r]).T

labels = ['No Pertenece','Pertenece']
NPertenece_R =len(y)-sum(y)
SPertenece_R = sum(y)
NPertenece_P = sum(pronosticos_R[0])
SPertenece_P = sum(pronosticos_R[1])
vals_P = [NPertenece_P, SPertenece_P]
vals_R = [NPertenece_R, SPertenece_R]
corde = np.arange(len(vals_P))
ancho = 0.3

fig, ax = plt.subplots()
ax.bar(corde-ancho/2, vals_P, ancho, label='Pronóstico')
ax.bar(corde+ancho/2, vals_R, ancho, label ='Real')

for i,j in zip(corde, vals_P):
        ax.annotate(j, xy=(i-0.2, j+0.2))
for i,j in zip(corde, vals_R):
        ax.annotate(j, xy=(i+0.1, j+0.2))

ax.set_title('Gráfico de Pronóstico v.s Real: Niño Mercado')
ax.set_xticks(corde)
ax.set_xticklabels(labels)

plt.legend();
```

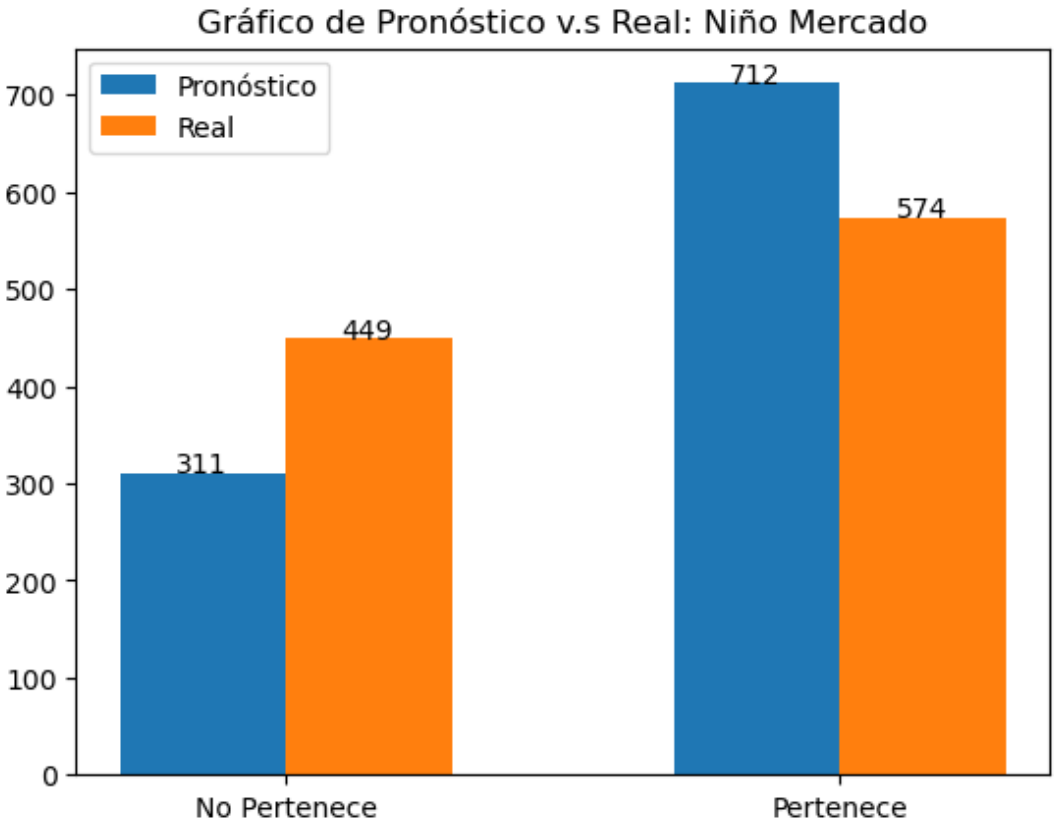

Figura 6.38: Gráfico de barras que muestra en el eje X si el individuo pertenece, o no, al Subprograma Niño Mercado. Cada barra muestra el total de individuos dentro de cada categoría segmentados por origen (real o pronóstico).

6.8. Pronóstico: Niño Trabajador

[5]:
```
df_R_nt=pd.read_excel('df_Tree_Gini_NT.xlsx', sheet_name='real')

X_tree_gini_fi=
df_R_nt[['REPORTE_CLASIFICACION_A','EDAD',
'ESCUELA_ASISTE','ENPROGRAMA',
'INGRESO_QUINCENAL']]

y = df_R_nt[['Niño Trabajador']]

y = y['Niño Trabajador'].tolist()

testsize = 1/3

depth = 3

X_e,X_p,y_e,y_p=
train_test_split(X_tree_gini_fi,y,test_size=testsize,stratify=y)

cad_gini = DecisionTreeClassifier(criterion =
'gini',max_depth=depth,random_state=0)
```

```
cad_gini.fit(X_e,y_e)
ypred_gini = cad_gini.predict(X_p)

y_pros_R= pd.DataFrame(cad_gini.predict_proba(X_tree_gini_fi))

vals_1_r = []
vals_2_r = []

for pros in y_pros_R[0]:
        if pros<=0.5:
                vals_1_r.append(0)
        else:
                vals_1_r.append(1)

for pros in y_pros_R[1]:
        if pros<=0.5:
                vals_2_r.append(0)
        else:
                vals_2_r.append(1)

pronosticos_R = pd.DataFrame([vals_1_r,vals_2_r]).T

labels = ['No Pertenece','Pertenece']
NPertenece_R =len(y)-sum(y)
SPertenece_R = sum(y)
NPertenece_P = sum(pronosticos_R[0])
SPertenece_P = sum(pronosticos_R[1])
vals_P = [NPertenece_P, SPertenece_P]
vals_R = [NPertenece_R, SPertenece_R]
corde = np.arange(len(vals_P))
ancho = 0.3

fig, ax = plt.subplots()
ax.bar(corde-ancho/2, vals_P, ancho, label='Pronóstico')
ax.bar(corde+ancho/2, vals_R, ancho, label ='Real')

for i,j in zip(corde, vals_P):
        ax.annotate(j, xy=(i-0.2, j+0.2))
for i,j in zip(corde, vals_R):
        ax.annotate(j, xy=(i+0.1, j+0.2))

ax.set_title('Gráfico de Pronóstico v.s Real: Niño Trabajador')
ax.set_xticks(corde)
ax.set_xticklabels(labels)

plt.legend();
```

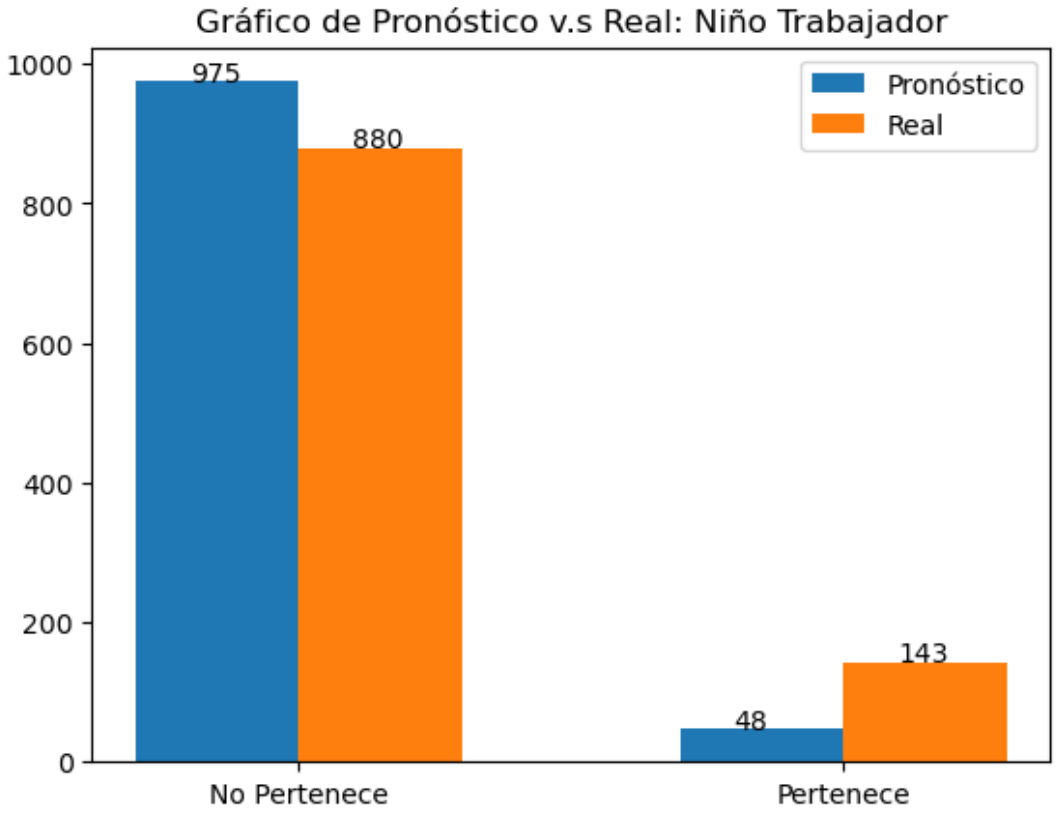

Figura 6.39: Gráfico de barras que muestra en el eje X si el individuo pertenece, o no, al Subprograma Niño Trabajador. Cada barra muestra el total de individuos dentro de cada categoría segmentados por origen (real o pronóstico).

Conclusión

En la era digital en la que vivimos, el aprendizaje automático y la inteligencia artificial (IA) han surgido como herramientas poderosas con el potencial de transformar una amplia gama de industrias y sectores. Si bien, tradicionalmente, se han asociado con aplicaciones en campos como la tecnología, la medicina y la industria; su relevancia y utilidad se extienden mucho más allá de estos dominios. Uno de los ámbitos donde su aplicación podría tener un impacto significativo y positivo es en el ámbito social, particularmente en el trabajo de las fundaciones cuyo interés es de tipo social sin fines de lucro.

Las fundaciones sin fines de lucro, cuyo interés es analizar problemas de tipo social, desempeñan un papel crucial en la sociedad al abordar diversas problemáticas; desde la pobreza, educación, hasta la salud y el medio ambiente. Sin embargo, la efectividad de su trabajo a menudo se ve limitada por los recursos, la complejidad de los problemas que enfrentan y la necesidad de tomar decisiones informadas para maximizar su impacto. Es aquí donde las tecnologías de aprendizaje automático e IA pueden ser de gran ayuda. Las fundaciones recopilan y analizan una gran cantidad de datos sobre las comunidades a las que sirven, desde datos demográficos hasta información sobre programas y resultados. El aprendizaje automático puede ayudar a estas organizaciones a extraer información significativa de estos datos, identificar patrones, predecir tendencias y tomar decisiones informadas basadas en evidencia, mediante el análisis de datos demográficos y socio económicos. Las fundaciones identifican áreas geográficas o poblaciones específicas que tienen mayores necesidades y dirigir sus recursos de manera más efectiva. Del mismo modo, el aprendizaje automático ayudan a evaluar la eficacia de los programas y estrategias implementadas por las fundaciones, identificando qué enfoques son más exitosos y dónde se existen áreas de

oportunidad.

En este caso de estudio en particular, se conoce el crecimiento de individuos dentro de los Subprogramas teniendo en cuenta solo seis características. Esto ayuda a segmentar de manera adecuada a la población para identificar cual Subprograma es el más idóneo para el individuo basándose en las características mencionadas al principio. Además de conocer a profundidad el comportamiento de cada Subprograma actual y en un futuro probable.

Otro aspecto importante a tomar en cuenta, es la capacidad de la IA para automatizar tareas repetitivas y optimizar procesos. Las fundaciones a menudo enfrentan una carga de trabajo abrumadora, desde la gestión de donaciones y la distribución de fondos; hasta la comunicación con los beneficiarios y la evaluación del impacto. Al aplicar herramientas de IA las fundaciones pueden liberar recursos humanos para tareas más estratégicas y centrarse en su misión principal.

En resumen, la aplicación del aprendizaje automático y la inteligencia artificial en áreas sociales tiene el potencial de generar un impacto significativo y positivo en la sociedad. Al aprovechar estas tecnologías, las fundaciones sin fines de lucro incrementarían la eficiencia de sus operaciones, aumentar la efectividad de sus programas y, en última instancia, contribuir de manera más efectiva a la resolución de los desafíos sociales más apremiantes de nuestro tiempo.

¿Por qué lucharías tanto para cumplir el sueño de otro?
Necesitas encontrar tu propio sueño.

Lucyna Kushinada (Lucy), Cyberpunk Edgerunners.

7

Conclusiones Generales

7.1. El Problema de Ser Adulto

Mediante el uso de distintos métodos e implementaciones, se identifican diferentes cuestiones respecto al uso de estas:

- **Modelos de regresión:**
  El uso de los modelos de regresión, para la implementación del aprendizaje automático, resulta muy acertada para problemáticas dónde el objetivo, o cuestión, se interpreta de manera sencilla. Problemas de comportamiento lineal, o de clasificación simple, son los mejores para ser atacados mediante los modelos de regresión. Incluso si este contiene muchas variables independientes para el análisis. La debilidad de estas implementaciones recae en la capacidad para predecir cuando el comportamiento del problema no es lineal o de clasificación simple. Ya que, como se vio en el capítulo correspondiente, existen métodos que clasifican de mejor manera y, por ende, generan una mejor predicción de los valores. Sin embargo, esto no deja mal parado a los modelos de regresión, ya que son una herramienta potente para, si no resolver en totalidad el problema, tener un primer acercamiento a este y conocer el comportamiento de los individuos para su correcto modelado.

- **k-Vecino más cercano:**
  En cuanto a métodos de clasificación, la metodología del k-vecino más cercano resulta muy efectiva al tomar en cuenta la *geometría*, en términos de distancia, de las instancias entre sí. El uso de agrupaciones para identificar características *parecidas* entre los individuos, además de sonar lógico, resulta bastante exacto al momento de generar predicciones o análisis de los individuos involucrados en la implementación. Por otro lado, un punto débil de esta metodología es el hecho de *¿Qué tan cercano es cercano?* ya que, tomar un número k de vecinos muy grande conlleva a caer en errores de sobre estimación. Es como querer decir que el shampoo y la mayonesa van en la misma estantería de la sala. Pero, teniendo cuidado a la hora de implementar y, sobre todo, conociendo a fondo la problemática a resolver, el método del k-vecino más cercano es una herramienta poderosa para clasificar mediante agrupaciones y, con ello, generar pronósticos de manera muy precisa.

- **Árboles de decisión:**
  La toma de decisiones es una parte fundamental de cualquier tipo de implementación.

Desde escoger las variables hasta aceptar o descartar hipótesis. Los árboles de decisión toman esta idea y la llevan al entorno computacional generando numerosas iteraciones para encontrar el mejor modelo. De todas, esta es la implementación más robusta al poder extender el análisis a un número grande de árboles de decisión (conocido como *bosques aleatorios)*; los cuales clasifican de una manera precisa al tomar en cuenta varias iteraciones y distintos tipos de evaluación (como el índice Gini y la entropía). Una cuestión puntual es que se debe tener bastante cuidado a la hora de generar árboles, ya que un árbol generado a partir de datos erróneos (o de baja calidad) generará un árbol deficiente a la hora de realizar predicciones. Por ello, este método es al que más se le debe de tener cuidado por la sensibilidad del mismo. Esto hace que sea un modelo de implementación refinado y, a su vez, una gran opción para realizar análisis con aprendizaje automático.

- **Caso de estudio:**
  Al encontrarse con los datos en bruto se contemplan dos grandes cuestiones:

  1. La calidad de los datos.
  2. La implementación de los modelos.

  En cuanto a la **calidad de los datos** siempre hay que llevar a cabo un proceso de preprocesamiento y preparación para que la implementación de cualquier modelo sea satisfactoria. Uno de los retos al trabajar con datos es justamente mejorar la calidad de los mismos sin intervenirlos directamente. Esto es aún más importante que la implementación misma; ya que, sin datos de buena calidad, aunque la implementación sea correcta, los resultados obtenidos serán deficientes y no contemplaran o resolverán la problemática principal.

  Por otro lado, la **implementación de los modelos** resulta un análogo a un juego de rompecabezas, solo que sin la imagen guía. Encontrar el mejor modelo para resolver la problemática siempre es un desafío, no solo por la búsqueda, sino por la interpretación final. En cuestión de analizar problemas de índole social, esto resulta más desafiante para la interpretación ya que todos los datos involucrados vienen de individuos reales. Por ende, las probabilidades y resultados generados deben de ser, forzosamente, pensados de manera profunda e interpretativa desde varios ángulos.

  En general, el implementar modelos de datos y aprendizaje automático a cuestiones de índole social no solo ayuda a tener un campo más de aplicación; ayuda a concientizarse un poco más con los individuos que, aunque no los conozcamos, nos hacemos una idea del contexto en el que viven. Y, sobre todo, cuestionarnos sobre la movilidad social y el cómo impactan las fundaciones sin fines de lucro a esta misma.

  Para finalizar, quiero invitar a la reflexión acerca del contexto social en el que vivimos y salirnos un poco de la burbuja para conocer realmente como es donde habitamos; eso es algo que el aprendizaje automático, ni la inteligencia artificial, pueden hacer. ¿Qué podemos esperar de un mundo que rechaza a los suyos sólo por nacer en un lugar diferente? y ¿qué podemos hacer para no llegar con vergüenza al futuro?.

Cazador, la noche ha llegado a su fin.

Gherman, Bloodborne.

8 Anexos

8.1. Catálogo de funciones

[ ]:
```
# Catálogo de Funciones

## INTRODUCCIÓN:
### Las funciones de python que se muestran son las que se utilizan a lo largo
### de los ejemplos y el caso de estudio.

## BIBLIOTECAS
### Se requieren instalar las siguientes paqueterías
#### pip install tabulate
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import warnings
warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
```

```
## FUNCIONES
### Descritivas
def head_describe_xlsx(path):

        data=pd.read_excel(path) #Carga de base de datos

        print(f'{data.head(5).to_markdown()} \n\n
         {data.describe().T.to_markdown()}')


def head_describe_csv(path):

        data=pd.read_csv(path) #Carga de base de datos

print(f'{data.head(5).to_markdown()} \n\n {data.describe().T.to_markdown()}')

### GRÁFICAS GENERALES
def MatrizCorrelación(
        data, #Base de datos
        size=(20,15), #Tamaño de la figura
        Tsize=25 #Tamaño del título
        ):

        plt.rcParams["figure.figsize"]=size #Tamaño de la figura

        corre=data.corr() #Creación de la matriz
        mask=np.triu(np.ones_like(corre,dtype=bool)) #Objeto para graficar

        sns.heatmap(corre,
        mask=mask,
        center=0,
        annot=True,
        fmt='.1f',
        square=True) #Mapa de calor

        plt.title("Matriz de Correlación",fontsize=Tsize) #Título de la figura
        plt.show() #Orden para mostrar la figura

def MatrizDiagramasDispersion(
        rows, #Número de filas
        cols, #Número de columnas
        X_col, #Nombres de las columnas de X
        y_col, #Nombre de la columna de y
        data, #Base de datos
        size=(20,15), #Tamaño de la figura
        Tsize=25, #Tamaño del título
        TSpsize=15, #Tamaño de los títulos de cada gráfico
        xlabsize=10, #Tamaño de etiquetas eje x
        ylabsize=10 #Tamaño de etiquetas eje y
        ):
```

```
        fig,ax=plt.subplots(nrows=rows,
        ncols=cols,
        figsize=size) #Creación de la figura
        ax_index=[] #Creación del index
        color=sns.color_palette("flare",cols*rows) #Color de las figuras

        for i in range(rows): #Orden del index
                for j in range(cols):
                        ax_index.append(ax[i,j])

        for k in range(len(X_col)): #Creación y orden de cada gráfico
                ax_index[k].scatter(
                x=data[X_col[k]],
                y=data[y_col],
                color=color[k],
                linewidth=1,
                alpha=0.5)
                ax_index[k].set_title(
                f'Dispersión del {y_col[0]}vs{X_col[k]}',
                fontsize=TSpsize) #Títulos
                ax_index[k].set_xlabel(f'{X_col[k]}',
                fontsize=xlabsize) #Etiquetas en eje x
                ax_index[k].set_ylabel(f'{y_col[0]}',
                fontsize=ylabsize) #Etiquetas en eje y

        plt.tight_layout(pad=1,w_pad=1,h_pad=1,rect=[0,0,1,0.95]) #Sangría
        fig.suptitle('Matriz de diagramas de dispersión',
        fontsize=Tsize) #Título de la figura
        plt.show() #Orden para mostrar la figura

def MatrizCajasBigotes(
        rows, #Número de fila
        cols, #Número de columnas
        X, #Valores de X
        data, #Base de datos
        size=(20,15), #Tamaño de la figura
        Tsize=25, #Tamaño del título
        TSpsize=15, #Tamaño de los títulos de cada gráfico
        xlabsize=10, #Tamaño de etiquetas eje x
        Vsize=10 #Tamaño de etiquetas de los valores
        ):

        plt.rcParams["figure.figsize"]=size #Tamaño de la figura

        fig, ax=plt.subplots(rows, cols) #Creación de la figura
        ax_index=[] #Creación del index
        color=sns.color_palette('inferno',cols*rows) #Color de las figuras

        for i in range(rows): #Orden del index
                for j in range(cols):
```

```
                        ax_index.append(ax[i,j])

        for k in range(len(X.columns)): #Creación y orden de cada gráfico
                g=sns.boxplot(data, x=X[list(X)[k]],
                ax=ax_index[k], color=color[k])
                g.set_title(f'{X.columns[k]}', fontsize=TSpsize) #Títulos
                g.set_xlabel(X.columns[k],fontsize=xlabsize) #Etiquetas x
                g.tick_params(labelsize=Vsize) #Tamaño de los valores

        fig.suptitle(
        f'Matriz de Gráficos de Cajas y Bigotes',
        fontsize=Tsize) #Título de gráficos
        plt.tight_layout(pad=1, w_pad=1, h_pad=2,
        rect=[0, 0, 1, 0.95]) #Sangría
        plt.show() #Orden para mostrar la figura

def MatrizViolines(
        rows, #Número de fila
        cols, #Número de columnas
        X, #Valores de X
        y, #Valores de y
        data, #Base de datos
        size=(20,15), #Tamaño de la figura
        Tsize=25, #Tamaño del título
        TSpsize=15, #Tamaño de los títulos de cada gráfico
        xlabsize=10, #Tamaño de etiquetas eje x
        Vsize=10 #Tamaño de etiquetas de los valores
        ):

        plt.rcParams["figure.figsize"]=size #Tamaño de la figura

        fig, ax=plt.subplots(rows, cols) #Creación de la figura
        ax_index=[] #Creación del index
        color=sns.color_palette('inferno',cols*rows) #Color de las figuras

        for i in range(rows): #Orden del index
                for j in range(cols):
                        ax_index.append(ax[i,j])

        for k in range(len(X.columns)): #Creación y orden de cada gráfico
                g=sns.violinplot(data=data,
                x=X[list(X)[k]], ax=ax_index[k], color=color[k])
                g.set_title(f'{X.columns[k]}', fontsize=TSpsize) #Títulos
                g.set_xlabel(X.columns[k],fontsize=xlabsize) #Etiquetas x
                g.tick_params(labelsize=Vsize) #Tamaño de los valores

        fig.suptitle(
        f'Matriz de gráficos de violín con valores de X vs {y.columns[0]}',
        fontsize=Tsize) #Título de la figura
        plt.tight_layout(pad=0, w_pad=1, h_pad=1,
        rect=[0, 0, 1, 0.95]); #Sangría
```

```
def DistKDE(
        data, #Base de datos
        Tsize=25, #Tamaño del título
        ):

        g=sns.pairplot(data,
        diag_kind='kde', corner=True) #Creación del gráfico

        g.fig.suptitle(f'Matriz de Distribución Kernel de la Densidad',
        fontsize=Tsize) #Título del gráfico
        g.tight_layout(pad=0, w_pad=1,h_pad=1,
        rect=[0, 0, 1, 0.95]); #Sangría

def HistogramaKDE(
        data, #Base de datos
        y, #Valores de y
        size=(10,5), #Tamaño de la figura
        Tsize=25 #Tamaño del título
        ):

        plt.rcParams["figure.figsize"]=size #Tamaño de la figura

        g=sns.histplot(
        data['Resultado'],kde=True,stat='density',
        kde_kws=dict(cut=3))#Creación de la figura

        g.set_title(
        f'Frecuencia y densidad de los valores de{y.columns[0]}',
        fontsize=Tsize) #Título
        plt.tight_layout(pad=0, w_pad=1, h_pad=1,
        rect=[0, 0, 1, 1.2]) #Sangría
        plt.show() #Orden para mostrar la figura

def MVTestDisperGraf(
        X_col, #Columnas de X
        y_col, #Columnas de y
        X, #Valores de X
        y #Valores de y
        ):

        x_train, x_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/3, #Tamaño del conjunto de prueba
        random_state=0 #Generación de la semilla (seed)
        )

        plt.rcParams["figure.figsize"]=(25, 15) #Tamaño de la figura

        y_pred=[] #Lista de valores predecidos de y
```

```
        SN=[] #Suma del numerador
        SD=[] #Suma del denominador
        y_mean=y_test.mean() #Media de y
        x_mean=x_test.mean() #Media de x

        for i in range(0,len(x_test)): #Valores numerador y denominador
                SN.append((x_test[i]-x_mean)*(y_test[i]-y_mean))
                SD.append((x_test[i]-x_mean)**2)

        b_1=np.divide(sum(SN),sum(SD)) #Valor del coeficiente
        b_0=y_mean-b_1*x_mean #Valor del intercepto

        for j in range(0,len(y_test)): #Cálculo de valores pronosticados
                y= b_0 + b_1*x_test[j]
                y_pred.append(y[0])

        plt.scatter(x_test, y_test,
        color="red", label="Conjunto de Prueba") #Gráfico de dispersión
        plt.plot(x_test,y_pred,
        color="blue", label="Recta de Regresión") #Recta de regresión
        plt.title(
        f'{X_col[0]} vs {y_col[0]} (Conjunto de prueba)', fontsize=30) #Título
        plt.legend() #Leyendas de los datos
        plt.ylabel(f'{y_col[0]}', fontsize=20) #Etiquetas del eje y
        plt.xlabel(f'{X_col[0]}', fontsize=20) #Etiquetas del eje x
        plt.show() #Orden para mostrar la figura

def MVTrainDisperGraf(
        X_col, #Columnas de X
        y_col, #Columnas de y
        X, #Valores de X
        y #Valores de y
        ):

        x_train, x_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/3, #Tamaño del conjunto de prueba
        random_state=0 #Generación de la semilla (seed)
        )

        plt.rcParams["figure.figsize"]=(25, 15) #Tamaño de la figura

        y_pred=[] #Lista de valores predecidos de y
        SN=[] #Suma del numerador
        SD=[] #Suma del denominador
        y_mean=y_train.mean() #Media de y
        x_mean=x_train.mean() #Media de x

        for i in range(0,len(x_train)): #Valores numerador y denominador
                SN.append((x_train[i]-x_mean)*(y_train[i]-y_mean))
```

```
                SD.append((x_train[i]-x_mean)**2)

        b_1=np.divide(sum(SN),sum(SD)) #Valor del coeficiente
        b_0=y_mean-b_1*x_mean #Valor del intercepto

        for j in range(0,len(y_train)): #Cálculo de valores pronosticados
                y= b_0 + b_1*x_train[j]
                y_pred.append(y[0])

        plt.scatter(x_train, y_train,
        color="red", label="Conjunto de Entrenamiento") #Gráfico de dispersión
        plt.plot(x_train,y_pred,
        color="blue", label="Recta de Regresión") #Recta de regresión
        plt.title(
        f'{X_col[0]} vs {y_col[0]} (Conjunto de entrenamiento)',
        fontsize=30) #Título
        plt.legend() #Leyendas de los datos
        plt.ylabel(f'{y_col[0]}', fontsize=20) #Etiquetas del eje y
        plt.xlabel(f'{X_col[0]}', fontsize=20) #Etiquetas del eje x
        plt.show() #Orden para mostrar la figura

def MaxVerRLS(
        X, #Valores de X
        y  #Valores de y
        ):

        x_train, x_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/3, #Tamaño del conjunto de prueba
        random_state=0 #Generación de la semilla (seed)
        )

        SN=[] #Suma numerador
        SD=[] #Suma denominador
        y_pred=[] #Valores de y pronóstico
        V_total=[] #Variación total
        V_expl=[] #Variación explicada
        y_pred_format=[] #Formato de salida
        y_mean=y_train.mean() #Media de y
        x_mean=x_train.mean() #Media de X

        for i in range(0,len(x_train)):#Valores numerador y denominador
                SN.append((x_train[i]-x_mean)*(y_train[i]-y_mean))
                SD.append((x_train[i]-x_mean)**2)

        b_1=sum(SN)/sum(SD) #Valor del coeficiente
        b_0=y_mean-b_1*x_mean #Valor del intercepto

        for j in range(0,len(y_train)): #Cálculo de valores pronosticados
                y= b_0 + b_1*x_train[j]
```

```
                y_pred.append(y[0])

        for u in range(0,len(y_pred)): #Cálculo de variaciones
                V_total.append((y_train[u]-y_train.mean())**2)
                V_expl.append((y_pred[u]-y_train.mean())**2)

        R2_1=(np.array(V_expl).sum())/(np.array(V_total).sum())

        for yp in range(0,5): #Declarando el formato de salida
                y_pred_format.append(y_pred[yp])
                y_pred_format=list(y_pred_format)
                Output=["%.2f" % elem for elem in y_pred_format] #Salida

        print(
        f'Valor del intercepto: {b_0[0]:.2f} \n
        Valor del coeficiente: {b_1[0]:.2f}') # Valores de beta's
        print(
        f'Primeros cinco valores de la regresión: {Output}') # pronóstico
        print(
        f'Coeficiente de Determinación: {R2_1*100:.2f}%') # R^2


def AARLS(
        X, # Valores de X
        y, # Valores de y
        ):

        x_train, x_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/3, #Tamaño del conjunto de prueba
        random_state=0 #Generación de la semilla (seed)
        )
        model = LinearRegression() # Modelo de regresión simple
        model.fit(x_train, y_train) # Ajuste de los datos de entrenamiento
        y_pred= model.predict(x_train) # Cálculo de los valores pronóstico
        y_pred_format = [] # Formato de salida
        for yp in range(0,5):
                y_pred_format.append(y_pred[yp])

        y_pred_format = list(y_pred_format)
        Output = ["%.2f" % elem for elem in y_pred_format] # Salida de valores

        b_0 = model.intercept_[0] # Valor del intercepto de regresión
        b_1 = model.coef_[0][0] # Valor del coeficiente de regresión

        print(
        f'Valor del intercepto: {b_0:.2f} \n
        Valor del coeficiente: {b_1:.2f}') # Valores de beta's
        print(
        f'Primeros cinco valores de la regresión: {Output}') # pronóstico
```

```
        print(
        f'Coeficiente de Determinación:
        {model.score(x_train,y_train)*100:.2f}%')


def AATrainDisperGraf(
        X_col, #Columnas de X
        y_col, #Columnas de y
        X, #Valores de X
        y #Valores de y
        ):

        x_train, x_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/3, #Tamaño del conjunto de prueba
        random_state=0 #Aleatoriedad
        )

        plt.rcParams["figure.figsize"] = (25, 15) # Tamaño de la figura

        reg = LinearRegression() # Modelo de regresión simple
        reg.fit(x_train, y_train) # Ajuste de los datos de entrenamiento
        y_pred= reg.predict(x_train)

        plt.scatter(x_train, y_train,
        color = "red", label="Conjunto de Entrenamiento") #Dispersión
        plt.plot(x_train,y_pred ,
        color = "blue", label="Recta de Regresión") #Recta de regresión
        plt.title(
        f'{X_col[0]} vs {y_col[0]} (Conjunto de entrenamiento)',
        fontsize = 30) #Título
        plt.ylabel(f'{y_col[0]}', fontsize = 20) #Etiquetas del eje y
        plt.xlabel(f'{X_col[0]}', fontsize = 20) #Etiquetas del eje x
        plt.legend() #Leyendas de los datos
        plt.show() #Orden para mostrar la figura

def AATestDisperGraf(
        X_col, #Columnas de X
        y_col, #Columnas de y
        X, #Valores de X
        y #Valores de y
        ):

        x_train, x_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/3, #Tamaño del conjunto de prueba
        random_state=0 #Generación de la semilla (seed)
        )
```

```
        plt.rcParams["figure.figsize"] = (25, 15) # Tamaño de la figura

        reg = LinearRegression() # Modelo de regresión simple
        reg.fit(x_test, y_test) # Ajuste de los datos de entrenamiento
        y_pred= reg.predict(x_test) # Cálculo de los valores pronóstico

        plt.scatter(x_test, y_test, color = "red",
        label="Conjunto de Prueba") #Gráfico de dispersión
        plt.plot(x_test,y_pred, color = "blue",
        label="Recta de Regresión") #Recta de regresión
        plt.title(f'{X_col[0]} vs {y_col[0]} (Conjunto de Prueba)',
        fontsize = 30) #Título
        plt.ylabel(f'{y_col[0]}', fontsize = 20) #Etiquetas del eje y
        plt.xlabel(f'{X_col[0]}', fontsize = 20) #Etiquetas del eje x
        plt.legend() #Leyendas de los datos
        plt.show() #Orden para mostrar la figura

def ECM_RLS(
        X, # Valores de X
        y # Valores de y
        ):

        X_train, X_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/3, #Tamaño del conjunto de prueba
        random_state=0 #Generación de la semilla (seed)
        )

        reg = LinearRegression()
        reg.fit(X_train, y_train)
        y_pred_ML= reg.predict(X_train)

        SN=[] #Suma numerador
        SD=[] #Suma denominador
        y_predMV=[] #Valores de y pronóstico
        V_total=[] #Variación total
        V_expl=[] #Variación explicada
        y_mean=y_train.mean() #Media de y
        x_mean=X_train.mean() #Media de X

        for i in range(0,len(X_train)):
                SN.append((X_train[i]-x_mean)*(y_train[i]-y_mean))
                SD.append((X_train[i]-x_mean)**2)

        b_1=sum(SN)/sum(SD) #Valor del coeficiente
        b_0=y_mean-b_1*x_mean #Valor del intercepto

        for j in range(0,len(y_train)): #Cálculo de valores pronosticados
                y= b_0 + b_1*X_train[j]
                y_predMV.append(y[0])
```

```
        for u in range(0,len(y_predMV)): #Cálculo de variaciones
                V_total.append((y_train[u]-y_train.mean())**2)
                V_expl.append((y_predMV[u]-y_train.mean())**2)

        R2_1=(np.array(V_expl).sum())/(np.array(V_total).sum())

        mse_1 = mean_squared_error(y_train,y_pred_ML)
        mse_2 = mean_squared_error(y_train,y_predMV)
        print('============================================================')
        print('ECM Aprendizaje Automático: ',mse_1)
        print('------------------------------------------------------------')
        print('ECM Máxima Verosimilitud: ',mse_2)
        print('============================================================')
        print('Diferencia cuadráda: ',(mse_1-mse_2)**2)
        print('============================================================')



def MinCua(
        X, # Valores de X
        y # Valores de y
        ):

        X_train, X_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/3, #Tamaño del conjunto de prueba
        random_state=0 #Generación de la semilla (seed)
        )

        y_predMV = []

        XtX_ = np.linalg.inv(np.dot(np.transpose(X_train),X_train))
        Xty_ = np.dot(np.transpose(X_train),y_train)

        B = np.dot(XtX_,Xty_)
        for i in range(0,len(X_train)):
                y_predMV.append(np.dot(np.transpose(B),X_train[i]))

        y_predMV_2 = []

        for i in range(0, len(y_predMV)):
                y_predMV_2.append(y_predMV[i][0])

        y_predMV = np.array(y_predMV_2)

        B_m = []
        for l in range(0,len(B)):
                B_m.append(B[l][0])
```

```
        B_m = np.array(B_m)

        BFormat = []

        for yp in range(0,len(B_m)-1): #Declarando el formato de salida
                BFormat.append(B_m[yp])
                BFormat=list(BFormat)
                Output=["%.2f" % elem for elem in BFormat] #Formato de salida


        yy = y_predMV[0:10]

        YFormat = []

        for yp in range(0,len(yy)-1): #Declarando el formato de salida
                YFormat.append(yy[yp])
                YFormat=list(YFormat)
                Output2=["%.2f" % elem for elem in YFormat] #Formato de salida

        print(f'Matriz de valores de beta: \n {Output}')
        print(f'Primeros 10 valores de la regresión: \n {Output2}')
        #print(f'Tabla resumen de la regresión: \n {modelo.summary()}')


def AARLM(
        X, # Valores de X
        y # Valores de y
        ):

        X_train, X_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/3, #Tamaño del conjunto de prueba
        random_state=0 #Generación de la semilla (seed)
        )

        reg = LinearRegression()
        reg.fit(X_train, y_train)
        y_predML = reg.predict(X_train)
        y_p = []
        for l in range(0,len(y_predML)):
        y_p.append(y_predML[l][0])

        B = reg.coef_[0]

        B_m = []
        for l in range(0,len(B)):
                B_m.append(B[l])

        B_m = np.array(B_m)
        BFormat = []
```

```
        for yp in range(0,len(B_m)-1): #Declarando el formato de salida
                BFormat.append(B_m[yp])
                BFormat=list(BFormat)
                Output=["%.2f" % elem for elem in BFormat] #Formato de salida

        YFormat = []
        yy = y_p[0:10]
        for yp in range(0,len(yy)-1): #Declarando el formato de salida
                YFormat.append(yy[yp])
                YFormat=list(YFormat)
                Output2=["%.2f" % elem for elem in YFormat] #Formato de salida

        X_train = sm.add_constant(X_train, prepend=True)
        modelo = sm.OLS(endog = y_train, exog = X_train)
        modelo = modelo.fit()

        print(f'Matriz de valores de beta: \n {Output}')
        print(f'Primeros 10 valores de la regresión: \n {Output2}')
        print(f'Tabla resumen de la regresión: \n {modelo.summary()}')

def ECM_RLM(
        X, # Valores de X
        y # Valores de y
        ):
        #Aprendizaje automático
        X_train, X_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/3, #Tamaño del conjunto de prueba
        random_state=0 #Generación de la semilla (seed)
        )

        reg = LinearRegression()
        reg.fit(X_train, y_train)
        y_predML = reg.predict(X_train)

        y_predMV = []

        XtX_ = np.linalg.inv(np.dot(np.transpose(X_train),X_train))
        Xty_ = np.dot(np.transpose(X_train),y_train)

        B = np.dot(XtX_,Xty_)
        for i in range(0,len(X_train)):
                y_predMV.append(np.dot(np.transpose(B),X_train[i]))

        mse_1 = mean_squared_error(y_train,y_predML)
        mse_2 = mean_squared_error(y_train,y_predMV)
        difcua = (mse_1-mse_2)**2
        print('_____________________________________________________________')
        print(f'ECM Aprendizaje Automático: {mse_1:.2f}')
        print('-------------------------------------------------------------')
```

```
        print(f'ECM Mínimos Cuadrádos: {mse_2:.2f}')
        print('_____________________________________________________________')
        print('')
        print('_____________________________________________________________')
        print(f'Diferencia cuadráda: {difcua:.2f}')
        print('_____________________________________________________________')



def RegLog(
        X, # Valores de X
        y # Valores de y
        ):
        X_train, X_test, y_train, y_test=train_test_split(
        X,
        y,
        test_size=1/4, #Tamaño del conjunto de prueba
        random_state=0 #Generación de la semilla (seed)
        )
        sc = StandardScaler()
        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)
        classifier = LogisticRegression(random_state = 0)
        classifier.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)
        print(f' Valores de la regresión: \n {y_pred}')



def matrx_conf(
        X, # Valores de X
        y, # Valores de y
        model, # Modelo de implementación
        test_size, # Tamaño de prueba
        random_state, # Aleatoriedad
        size=(10,7) # Tamaño de la figura
        ):
        X_train, X_test, y_train, y_test = train_test_split(
        X,y,test_size=test_size,
        random_state=random_state, stratify=y)
        sc = StandardScaler()
        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        fig, ax = plt.subplots(figsize = size)
        C2 = confusion_matrix(y_test, y_pred)
        sns.heatmap(C2, annot = True, ax = ax, linewidths = 1,
        fmt = "g", annot_kws={'size': 15})
        ax.set_title('Matriz de Confusión', fontsize = 20)
        ax.set_xlabel('Valor Pronóstico', fontsize = 15)
```

```
        ax.set_ylabel('Valor Prueba', fontsize = 15);
        VP = C2[0][0]
        VN = C2[0][1]
        FP = C2[1][1]
        FN = C2[1][0]
        print(f"""Total de Aciertos: {VP+FP}
                  \n Total de Fracasos: {VN+FN}
                  \n Proporción de Aciertos: {(VP+FP)/(VP+FP+VN+FN):.2%}""")

def ROC(
        X, # Valores de X
        y, # Valores de y
        model, # Modelo de implementación
        test_size, # Tamaño de prueba
        random_state, # Aleatoriedad
        size = (20, 15) # Tamaño de la figura
        ):
        X_train, X_test, y_train,
        y_test =
        train_test_split(X,y,
        test_size=test_size,random_state=random_state, stratify=y)
        sc = StandardScaler()
        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)
        model.fit(X_train, y_train)
        plt.rcParams["figure.figsize"] = size
        ns_probs = [0 for _ in range(len(y_test))]
        lr_probs = model.predict_proba(X_test)
        lr_probs = lr_probs[:, 1]
        ns_auc = roc_auc_score(y_test, ns_probs)
        lr_auc = roc_auc_score(y_test, lr_probs)
        print('Sin entrenar: ROC AUC=%.3f' % (ns_auc))
        print('Regresión Logística: ROC AUC=%.3f' % (lr_auc))
        ns_fpr, ns_tpr, _ = roc_curve(y_test, ns_probs)
        lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_probs)
        plt.plot(ns_fpr, ns_tpr, linestyle='--', label='Sin entrenar')
        plt.plot(lr_fpr, lr_tpr, marker='.', label='Regresión Logística')
        plt.title('Curva de ROC', fontsize = 20)
        plt.xlabel('Tasa de Falsos Positivos', fontsize = 15)
        plt.ylabel('Tasa de Verdaderos Positivos', fontsize = 15)
        plt.legend()
        plt.show()


def kde_plt_hue(
        data, # Base de datos
        y # Valores de y
        ):
        g = sns.pairplot(data, hue='Resultado', corner = True)
        g.fig.suptitle(
        f'Matriz de Distribución Kernel de la Densidad \n
```

```
        Categorizada con {y.columns[0]}', fontsize = 25)
        g.fig.set_size_inches(15,15)
        g.tight_layout();

def kde_plt_subset_hue(
        data, # Base de datos
        args # Argumentos
        ):
        g = sns.pairplot(data,
        hue = args[2],
        x_vars = args[1],
        y_vars = args[0]);
        g.fig.suptitle(
        f'Dispersión de
        {args[0]} y {args[1]} categorizada con {args[2]}',
        fontsize = 25)
        g.fig.set_size_inches(15,15)
        g.tight_layout();

def kde_plt_hue_cat(
        data, # Base de datos
        y # Argumentos
        ):
        plt.rcParams["figure.figsize"] = (25, 20)
        g = sns.pairplot(data,
        hue = y.columns[0], diag_kind="kde", corner=True)
        g.map_lower(sns.kdeplot, levels=1, color=".2")
        g.fig.suptitle(
        f'Matriz de Distribución Kernel de la Densidad \n
         Categorizada y agrupada con {y.columns[0]} ',
        fontsize = 25)
        g.fig.set_size_inches(15,15)
        g.tight_layout();

def heuristico_knn(
        maxK, # Valores de k
        X, # Valores de X
        y, # Valores de y
        test_size=1/4, # Tamaño de prueba
        random_state=0 # Aleatoriedad
        ):
        X_train, X_test, y_train, y_test = train_test_split(X,y,
        test_size=test_size,random_state=random_state,
        stratify=y)
        plt.rcParams["figure.figsize"] = (25, 20)
        test_p = []; train_p = []; ks=[]

        for k in range(1,maxK):
                model = KNeighborsClassifier(k)
                model.fit(X_train,y_train)
                train_p.append(model.score(X_train,y_train))
```

```
                test_p.append(model.score(X_test,y_test))
                ks.append(k)

        max_train_p = max(train_p)
        train_p_ind = [i for i, v in enumerate(train_p) if v == max_train_p]
        print(
        'Puntuación máxima de entrenamiento {} con k = {}'.format(
        max_train_p,list(map(lambda x: x+1, train_p_ind))))

        max_test_p = max(test_p)
        test_p_ind = [i for i, v in enumerate(test_p) if v == max_test_p]
        print(
        'Puntuación máxima de prueba {} con k = {}'.format(
        max_test_p,list(map(lambda x: x+1, test_p_ind))))

        raw=pd.DataFrame([ks,test_p,train_p]).T
        raw.columns=["ks","test","train"]

        test_p =raw["test"]
        train_p =raw["train"]
        p=sns.lineplot([test_p, train_p], legend="full")
        plt.title(f"""Gráfico de línea múltiple
            \n Comportamiento del conjunto de prueba y entrenamiento
            utilizando diferentes valores de vecinos (k={maxK})""",
        fontsize = 20);



def heuristico_arbol(
        depth, # Profundidad
        criterion, # Criterio
        X, # Valores de X
        y, # Valores de y
        test_size=1/4, # Tamaño de prueba
        rs=0 # Aleatoriedad
        ):

        X_e, X_p, y_e, y_p = train_test_split(
        X, y,
        test_size = test_size, stratify=y)
        prueba_puntaje = []; entre_puntaje = []
        for a in range(1,depth):
                cad_ent = DecisionTreeClassifier(criterion = criterion,
                max_depth = a, random_state = rs)
        cad_ent.fit(X_e, y_e)
        prueba_puntaje.append(cad_ent.score(X_p, y_p))
        entre_puntaje.append(cad_ent.score(X_e, y_e))
        ep_ind =
        [i for i, v in enumerate(entre_puntaje) if v == max(entre_puntaje)]
        print(
        'Puntuación máxima de entrenamiento
```

```
         con p = {}'.format(max(entre_puntaje),
        list(map(lambda x: x+1, ep_ind))))
        pp_ind =
        [i for i, v in enumerate(prueba_puntaje) if v == max(prueba_puntaje)]
        print('Puntuación máxima de prueba {} con p = {}'.format(
        max(prueba_puntaje), list(map(lambda x: x+1, pp_ind))))
        plt.figure(figsize = (12,5))
        sns.lineplot(x = range(1,depth), y=entre_puntaje,
        marker='*',
        label = 'Puntaje de Entrenamiento')
        sns.lineplot(x = range(1,depth),
        y = prueba_puntaje, marker = 'o',
        label = 'Puntaje de Prueba');


def Arbol(
        X, # Valores de X
        y, # Valores de y
        etiquetas, # Etiquetas
        testsize, # Tamaño de prueba
        criterion, # Criterio
        depth, # Profundidad
        rs, # Aleatoriedad
        Tsize=(12,12), # Tamaño de la figura
        ):

        X_e,X_p,y_e,y_p=train_test_split(
        X.values,y.values,
        test_size=testsize,stratify=y)
        CriterioClasif = DecisionTreeClassifier(
        criterion = criterion,max_depth=depth,random_state=rs)
        CriterioClasif.fit(X_e,y_e)
        ypred_gini = CriterioClasif.predict(X_p)
        print(
        f'Árbol de decisión \n Criterio: {criterion}. \n
         Profundidad: {depth} \n')
        print('Estadísticos: \n
         Precisión:{0:0.4f}'.format(accuracy_score(y_p, ypred_gini)))
        print(
        'Puntuación Conjunto de Entrenamiento:
        {:.4f}'.format(CriterioClasif.score(X_e,y_e)))
        print(
        'Puntuación Conjunto de Prueba:
        {:.4f}'.format(CriterioClasif.score(X_p, y_p)))
        print(
        'Diferencia Absoluta de la Puntuación de los Conjuntos:
        {0:0.4f}'.format(
        abs(CriterioClasif.score(X_e, y_e) - CriterioClasif.score(X_p, y_p))))
        fig, ax = plt.subplots(nrows = 2, ncols = 1, figsize = Tsize)
        tree.plot_tree(CriterioClasif.fit(X_e, y_e),ax = ax[0])
        importancia = CriterioClasif.feature_importances_
```

```
        indices = np.argsort(importancia)
        plt.title('Importancia de Características')
        ax[1] = plt.barh(range(len(indices)),
        importancia[indices], color = 'b',
        align = 'center')
        plt.yticks(range(len(indices)), [etiquetas[i] for i in indices])
        plt.xlabel('Importancia Relativa')
        fig.tight_layout()

def MatrizViolinesHue(
        rows, #Número de fila
        cols, #Número de columnas
        X, #Valores de X
        y, #Valores de y
        data, #Base de datos
        h, # Hue
        v, # Split
        size=(20,15), #Tamaño de la figura
        Tsize=25, #Tamaño del título
        TSpsize=15, #Tamaño de los títulos de cada gráfico
        xlabsize=10, #Tamaño de etiquetas eje x
        Vsize=10 #Tamaño de etiquetas de los valores
        ):

        plt.rcParams["figure.figsize"]=size #Tamaño de la figura

        fig, ax=plt.subplots(rows, cols) #Creación de la figura
        ax_index=[] #Creación del index
        color=sns.color_palette('inferno',cols*rows) #Color de las figuras

        for i in range(rows): #Orden del index
                for j in range(cols):
                        ax_index.append(ax[i,j])

        for k in range(len(X.columns)): #Creación y orden de cada gráfico
                g=sns.violinplot(
                data=data, x=X[list(X)[k]], hue = h,
                split = v, ax=ax_index[k], color=color[k])
                g.set_title(f'{X.columns[k]}', fontsize=TSpsize) #Títulos
                g.set_xlabel(X.columns[k],fontsize=xlabsize) #Etiquetas x
                g.tick_params(labelsize=Vsize) #Tamaño de los valores

        fig.suptitle(
        f'Matriz de gráficos de violín con valores de X vs
        {y.columns[0]} Clasificado con {h}',
        fontsize=Tsize) #Título de la figura
        plt.tight_layout(pad=0, w_pad=1, h_pad=1,
        rect=[0, 0, 1, 0.95]); #Sangría

def Violin(
        data, # Base de datos
```

```
        X_col, # Columnas de X
        y_col, # Columnas de y
        size=(20,15) # Tamaño de la figura
        ):
        plt.rcParams["figure.figsize"]=size
        sns.violinplot(data = data,
        x = X_col,
        y = y_col,
        palette = 'inferno').set(title = 'Insulina Agrupada por Resultado');


def ViolinHue(
        data, # Base de datos
        X_col, # Columnas de X
        y_col, # Columnas de y
        h,size=(20,15) # Tamaño de la figura
        ):
        plt.rcParams["figure.figsize"]=size
        sns.violinplot(data = data,
        x = X_col,
        y = y_col,
        hue = h ,
        palette = 'inferno').set(
        title =f'{y_col} Agrupada por {X_col} y Clasificado con {h}');


def ViolinHueSplit(
        data, # Base de datos
        X_col, # Columnas de X
        y_col, # Columnas de y
        h, # Hue
        v, # Split
        size=(20,15) # Tamaño de la figura
        ):
        plt.rcParams["figure.figsize"]=size
        sns.violinplot(data = data,
        x = X_col,
        y = y_col,
        hue = h,
        palette = "inferno",
        split = v).set(
        title =
        f'{y_col} Agrupada por {X_col} y Clasificado con {h} Seccionado');

def MatrixViolinesHueSplit(
        data, # Base de datos
        X, # Columnas de X
        y, # Columnas de y
        h, # Hue
        v=True, # Split
        rows=4, # Filas
```

```
        cols=3, # Columnas
        size = (22,11) # Tamaño de la figura
        ):
        plt.rcParams["figure.figsize"]=size
        fig, ax=plt.subplots(rows, cols) #Creación de la figura
        ax_index=[] #Creación del index
        color=sns.color_palette('inferno',cols*rows) #Color de las figuras

        for i in range(rows): #Orden del index
                        for j in range(cols):
                                ax_index.append(ax[i,j])
                                labels=X.columns
        for k in range(len(X.columns)): #Creación y orden de cada gráfico
                g=sns.violinplot(data=data, y=labels[k],
                x=y.columns[0], hue = h,
                split = v, ax=ax_index[k], color=color[k])
                g.set_title(f'{X.columns[k]}', fontsize=20) #Títulos
                g.set_xlabel(y.columns[0],fontsize=15) #Etiquetas en eje x
                g.set_ylabel(X.columns[k], fontsize = 15) #Etiquetas en eje y
                g.tick_params(labelsize=10) #Tamaño de los valores

                fig.suptitle(
                f'Matriz de gráficos de violín con valores de X vs
                {y.columns[0]} Clasificado con {h}',
                fontsize=10) #Título de la figura
                plt.tight_layout(pad=0, w_pad=1, h_pad=1,
                rect=[0, 0, 1, 0.95]); #Sangría
```

Bibliografía

[1] Autores, V. (1990). Diabetes dataset. https://www.kaggle.com/datasets/mathchi/diabetes-data-set.

[2] Azevedo, A. and Santos, M. F. (2008). Kdd, semma and crisp-dm: a parallel overview. *IADS-DM*.

[3] Barker, R. (1994). *El modelo entidad-relación CASE* methodtm*. Ediciones Díaz de Santos.

[4] Boden, M. A. (2017). *Inteligencia artificial*. Turner.

[5] Boehmke, B. and Greenwell, B. (2019). *Hands-on machine learning with R*. Chapman and Hall/CRC.

[6] Casella, G. and Berger, R. L. (2021). *Statistical inference*. Cengage Learning.

[7] de la Diabetes y las Enfermedades Digestivas y Renales, I. N. (2022). https://www.niddk.nih.gov/.

[8] de México, G. (2022). Etnografía del pueblo pima.

[9] Forero-Corba, W., Bennasar, F. N., et al. (2023). Técnicas y aplicaciones del machine learning e inteligencia artificial en educación: una revisión sistemática. *RIED-Revista Iberoamericana De Educación a Distancia*, 27(1):209–253.

[10] Hernández-Orallo, J. (2013). ROC curves for regression. *Pattern Recognition*, 46(12):3395–3411.

[11] Hinestroza Ramírez, D. (2018). El machine learning a través de los tiempos, y los aportes a la humanidad.

[12] Japkowicz, N. and Shah, M. (2011). *Evaluating learning algorithms: a classification perspective*. Cambridge University Press.

[13] JUCONI, F. (2022). *JUCONI Reporte Anual 2022*. Fundación JUCONI.

[14] kaggle.com (2022). Bodyfat dataset. https://www.kaggle.com/datasets/fedesoriano/body-fat-prediction-dataset.

[15] Kotsiantis, S. B. (2013). Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283.

[16] Krause, E. F. (1986). *Taxicab geometry: An adventure in non-Euclidean geometry*. Courier Corporation.

[17] Lorenzo, J. (2019). Distancia de Mahalanobis.

[18] Moreno, A. (1994). *Aprendizaje Automático*. Ediciones UPC.

[19] Müller, A. C. and Guido, S. (2016). *Introduction to machine learning with Python: a guide for data scientists*. O'Reilly Media, Inc.

[20] OMS (2023). La OMS esboza las cuestiones que cabe tener en cuenta a fin de regular la inteligencia artificial para la salud. *OMS Comunicados de Prensa*.

[21] Portillo, M. T. E. and Plata, J. A. S. (2008). P. ch. Mahalanobis y las aplicaciones de su distancia estadística. *CULCyT: Cultura Científica y Tecnológica*, 5(27):13–20.

[22] Rahwan, I., Cebrian, M., Obradovich, N., Bongard, J., Bonnefon, J.-F., Breazeal, C., Crandall, J. W., Christakis, N. A., Couzin, I. D., Jackson, M. O., et al. (2019). Machine learning behaibor. *Nature*, 568(7753):477–486.

[23] Reyes, E. (1983). Métricas que pueden inducirse por una norma. *Revista Integración, temas de matemáticas*, 2(1):53–58.

[24] Reynolds, B. E. (1980). Taxicab geometry. *Pi Mu Epsilon Journal*, 7(2):77–88.

[25] Richert, W. (2013). *Building machine learning systems with Python*. Packt Publishing Ltd.

[26] Rincón, L. (2019). Una introducción a la estadística inferencial. *México: Universidad Nacional Autónoma de México*.

[27] Rokach, L. and Maimon, O. (2005). Decision trees. *Data mining and knowledge discovery handbook*, pages 165–192.

[28] Ryan, T. P. (2008). *Modern regression methods*, volume 655. John Wiley & Sons.

[29] Sagan, C. and Udina, D. (1997). *El mundo y sus demonios*. Planeta Barcelona.

[30] Sagiroglu, S. and Sinanc, D. (2013). Big data: A review. pages 42–47.

[31] Sakamoto, Y., Ishiguro, M., and Kitagawa, G. (1986). Akaike information criterion statistics. *Dordrecht, The Netherlands: D. Reidel*, 81(10.5555):26853.

[32] Sarkar, D., Bali, R., and Sharma, T. (2018). *Practical machine learning with Python*. Springer.

[33] Shao, J. (2003). *Mathematical statistics*. Springer Science & Business Media.

[34] Silberschatz, A., Korth, H. F., Sudarshan, S., Pérez, F. S., Santiago, A. I., and Sánchez, A. V. (2002). *Fundamentos de bases de datos*, volume 11. McGraw-Hill Ciudad de México, México.

[35] Van Der Aalst, W. (2012). Process mining. *Communications of the ACM*, 55(8):76–83.

[36] VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data*. O'Reilly Media, Inc.

[37] Wooldridge, J. M. (2006). *Introducción a la econometría. Un enfoque moderno: un enfoque moderno*. Ediciones Paraninfo, SA.

[38] Álvaro Lozano (2022). La historia del machine learning. https://elearningactual.com/la-historia-del-machine-learning/#:~:text=El%20tÃ©rmino%20âĂœAprendizaje%20AutomÃąticoâĂİ%20se,sin%20necesidad%20de%20intervenciÃ§n%20humana.&text=El%20nuevo%20milenio%20fue%20testigo,precedentes%20de%20la%20programaciÃ§n%20adaptativa.