**Unidad 2 - Tarea 3 - Algoritmos de Aprendizaje Supervisado**

**Nombre**

**VICTOR ALFONSO ARTUNDUAGA HENAO**

**Grupo (202016908\_47)**

**Tutor:**

**HANDRY OROZCO**

**Universidad Nacional Abierta y a Distancia UNAD**

**ANÁLISIS DE DATOS**

**2024**

**Introducción actividad**

La actividad consiste en el desarrollo de diversos ítems enfocados en el aprendizaje automático y el uso de modelos predictivos. En primer lugar, se llevará a cabo una revisión bibliográfica de la Unidad 2. Posteriormente, se elaborará un cuadro sinóptico que sintetice los diferentes modelos de aprendizaje supervisado, incluyendo su definición, casos de uso, ventajas y desventajas.

Se procederá a compilar un listado de definiciones clave, como Datos de Train, Datos de Validation y Test, GridSearchCV, One Hot Encoding, y métricas de evaluación como Precision, Accuracy, Specificity, Recall, F1 Score y curva ROC. El lenguaje utilizado será Python, empleando Jupyter notebooks, lo que requerirá la instalación de Anaconda, una distribución esencial para la ciencia de datos y el aprendizaje automático. Además, se descargarán datasets de la plataforma Kaggle, los cuales incluirán información sobre vehículos, enfermedades cardíacas y calidad del vino. Con estos conjuntos de datos, se diseñarán modelos predictivos de regresión lineal, regresión logística y árboles de decisión. Para cada modelo, se realizará un análisis exploratorio de los datos, se llevará a cabo el preprocesamiento necesario, se seleccionarán las características más relevantes, se dividirá el dataset en conjuntos de entrenamiento y prueba, y se entrenará el modelo ajustando los hiperparámetros.

Finalmente, se evaluará el desempeño de los modelos mediante métricas como precisión, recall y F1 Score, y se generarán gráficas para visualizar los resultados. La actividad también incluye la interpretación y documentación de los resultados obtenidos, así como la creación de una cuenta en GitHub para cargar los trabajos realizados.

**Justificación actividad**

La justificación de esta actividad radica en la creciente importancia del aprendizaje automático en diversos campos, desde la medicina hasta el marketing y la ingeniería. Al desarrollar habilidades prácticas en el uso de modelos predictivos, se facilita la comprensión de conceptos fundamentales y se promueve la aplicación efectiva de técnicas avanzadas de análisis de datos. La revisión bibliográfica de la Unidad 2 proporciona el contexto teórico necesario para comprender los fundamentos de los modelos de aprendizaje supervisado, permitiendo a los participantes familiarizarse con sus aplicaciones y limitaciones. La elaboración del cuadro sinóptico ofrece una forma visual de organizar y sintetizar esta información, facilitando la comparación entre diferentes enfoques.

El uso de definiciones clave relacionadas con el aprendizaje automático contribuye a establecer un vocabulario común que es esencial para la comunicación efectiva en este ámbito. La instalación de Anaconda y el uso de Jupyter notebooks son pasos cruciales para trabajar de manera eficiente con Python, una de las herramientas más utilizadas en ciencia de datos. La descarga y análisis de datasets de Kaggle permite aplicar los conceptos aprendidos en situaciones del mundo real, brindando una experiencia práctica invaluable. A través del análisis exploratorio, el preprocesamiento de datos y la evaluación de modelos, se fomenta un enfoque analítico que es esencial para la toma de decisiones informadas basadas en datos.

Por último, la interpretación y documentación de los resultados no solo permite a los participantes consolidar su aprendizaje, sino que también promueve la habilidad de comunicar hallazgos de manera clara y efectiva. La creación de una cuenta en GitHub para cargar los trabajos realizados añade un componente de colaboración y visibilidad a sus proyectos, lo que es fundamental en el entorno profesional actual. En resumen, esta actividad no solo refuerza el conocimiento teórico, sino que también desarrolla habilidades prácticas y comunicativas necesarias en el campo del aprendizaje automático.

**Objetivo general actividad**

Desarrollar habilidades prácticas en el ámbito del aprendizaje automático mediante la implementación de modelos predictivos utilizando Python y Jupyter notebooks, a partir de la revisión teórica y la aplicación de técnicas de análisis de datos en conjuntos de datos reales.

**Objetivos específicos actividad**

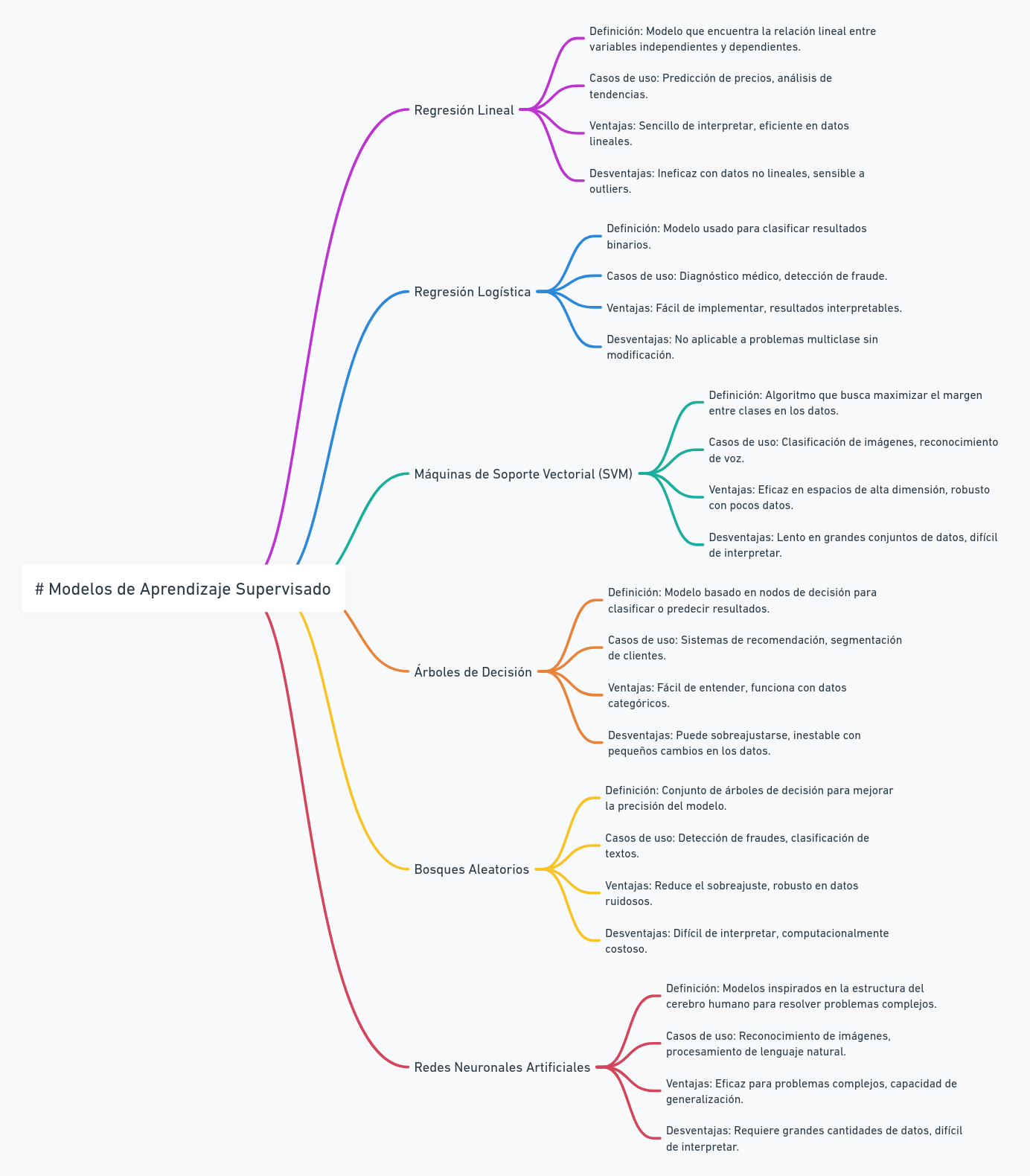
1. Revisar y sintetizar conceptos clave sobre aprendizaje supervisado a través de la revisión bibliográfica de la Unidad 2 y la elaboración de un cuadro sinóptico que facilite la comprensión de diferentes modelos y sus características.
2. Definir términos fundamentales relacionados con el aprendizaje automático, como Datos de Train, Datos de Validation y Test, y métricas de evaluación, para establecer un vocabulario técnico adecuado que permita una mejor comunicación y entendimiento en el campo.
3. Implementar modelos predictivos de regresión lineal, regresión logística y árboles de decisión utilizando datasets descargados de Kaggle, aplicando técnicas de análisis exploratorio, preprocesamiento de datos, selección de características y evaluación del desempeño del modelo.
4. Documentar e interpretar los resultados obtenidos a partir de los modelos implementados, generando gráficas que visualicen el desempeño y permitiendo una reflexión crítica sobre el proceso y los hallazgos, así como compartir los proyectos en una cuenta de GitHub.

**Desarrollo**

**• Elaborar un cuadro sinóptico sobre los diferentes modelos de Aprendizaje Supervisado que incluya definición, casos de uso, ventajas y desventajas.**

El mapa mental sobre los Modelos de Aprendizaje Supervisado organiza seis modelos clave: Regresión Lineal, Regresión Logística, Máquinas de Soporte Vectorial (SVM), Árboles de Decisión, Bosques Aleatorios y Redes Neuronales Artificiales. Cada uno incluye una breve definición, donde se describe su funcionamiento básico: la Regresión Lineal se enfoca en relaciones entre variables continuas, mientras que la Regresión Logística clasifica resultados binarios.

Las SVM buscan maximizar márgenes entre clases, los Árboles de Decisión utilizan nodos para decisiones, los Bosques Aleatorios combinan árboles para mayor precisión, y las Redes Neuronales se basan en estructuras inspiradas en el cerebro humano para resolver problemas complejos. Los casos de uso varían desde predicciones económicas hasta clasificación de imágenes y detección de fraudes. Entre las ventajas, se destacan la facilidad de interpretación de algunos modelos como la Regresión, mientras que las Redes Neuronales y Bosques Aleatorios son útiles para problemas más complejos. Sin embargo, los modelos más avanzados tienden a ser más costosos en términos computacionales y menos interpretables, en contraste con la simplicidad de la Regresión y los Árboles de Decisión.



**• Elaborar un listado con las siguientes definiciones: Datos de Train, Datos de Validation y Test, GridSearchCV, One Hot Encoding, Matriz de confusión, Precision, Accuracy, Specifiticy, Recall, F1 Score, curva ROC, R cuadrado.**

**Definiciones de Términos en Machine Learning**

**1. Datos de Train (Entrenamiento)**

Son los datos que se utilizan para entrenar un modelo de machine learning. El modelo ajusta sus parámetros a partir de estos datos, aprendiendo patrones subyacentes.

**2. Datos de Validation (Validación)**

Estos datos se utilizan para evaluar el modelo durante el proceso de entrenamiento y ajustar hiperparámetros. La validación asegura que el modelo no esté sobreajustando (overfitting) los datos de entrenamiento.

**3. Datos de Test (Prueba)**

Son un conjunto independiente de datos utilizados para evaluar el rendimiento final del modelo. Estos datos no se han visto durante el entrenamiento ni la validación, proporcionando una estimación realista de cómo el modelo generaliza a datos nuevos.

**4. GridSearchCV**

Es una técnica utilizada para encontrar la mejor combinación de hiperparámetros para un modelo. Busca de manera exhaustiva en un espacio definido de parámetros utilizando validación cruzada (cross-validation) para seleccionar la configuración que maximice el rendimiento del modelo.

**5. One Hot Encoding**

Es una técnica de preprocesamiento de datos categóricos, en la cual se transforman las variables categóricas en vectores binarios. Cada categoría única  
se convierte en una columna nueva y se asigna un valor de 1 en la columna correspondiente y 0 en las demás.

**6. Matriz de Confusión**

Es una tabla que se utiliza para evaluar el rendimiento de un modelo de clasificación. La matriz muestra la relación entre las predicciones reales y las predicciones hechas por el modelo, separando los valores correctamente clasificados de los errores.

* Verdaderos Positivos (VP): casos correctamente clasificados como positivos.
* Falsos Positivos (FP): casos incorrectamente clasificados como positivos.
* Verdaderos Negativos (VN): casos correctamente clasificados como negativos.
* Falsos Negativos (FN): casos incorrectamente clasificados como negativos.

**7. Precision (Precisión)**

Es una métrica que mide la exactitud de las predicciones positivas del modelo. Se calcula como la proporción de verdaderos positivos sobre todos los casos que el modelo predijo como positivos.

**8. Accuracy (Exactitud)**

Es la proporción de predicciones correctas (tanto positivas como negativas) sobre el total de casos. Es una métrica global del rendimiento del modelo.

**9. Specificity (Especificidad)**

Es la proporción de verdaderos negativos correctamente identificados por el modelo. Mide qué tan bien el modelo evita clasificar erróneamente los casos negativos como positivos.

**10. Recall (Sensibilidad o Tasa de Verdaderos Positivos)**

Es la proporción de verdaderos positivos correctamente identificados por el modelo, también conocido como sensibilidad. Mide qué tan bien el modelo detecta casos positivos.

**11. F1 Score**

Es la media armónica entre la precisión (precision) y el recall. Es útil cuando hay un desequilibrio entre clases, ya que considera tanto los falsos positivos como los falsos negativos.

**12. Curva ROC (Receiver Operating Characteristic)**

Es una gráfica que muestra la relación entre el Tasa de Verdaderos Positivos (Recall) y la Tasa de Falsos Positivos a diferentes umbrales de decisión del modelo. La curva ROC permite evaluar la capacidad del modelo para distinguir entre clases.

**13. cuadrado (Coeficiente de Determinación)**

Es una métrica utilizada en modelos de regresión que indica qué proporción de la variabilidad en la variable dependiente puede ser explicada por el mod-  
elo. Un valor de cercano a 1 indica que el modelo explica bien los datos.

|  |  |
| --- | --- |
| Término | Definición |
| Datos de Train | Datos utilizados para entrenar el modelo, permitiéndole aprender patrones subyacentes. |
| Datos de Validation | Datos utilizados para ajustar hiperparámetros y prevenir el sobreajuste (overfitting). |
| Datos de Test | Datos independientes utilizados para evaluar el rendimiento final del modelo. |
| GridSearchCV | Técnica para encontrar la mejor combinación de hiperparámetros usando validación cruzada (crossvalidation). |
| One Hot Encoding | Transformación de variables categóricas en vectores binarios, donde cada categoría es representada por una columna. |
| Matriz de Confusión | Tabla que muestra la relación entre las predicciones reales y las predicciones del modelo. |
| Precision (Precisión) | Proporción de verdaderos positivos sobre todos los casos predichos como positivos. |
| Accuracy (Exactitud) | Proporción de predicciones correctas sobre el total de casos. |
| Specificity (Especificidad) | Proporción de verdaderos negativos correctamente identificados. |
| Recall (Sensibilidad) | Proporción de verdaderos positivos correctamente identificados. |
| F1 Score | Media armónica entre precisión y recall. |
| Curva ROC | Gráfica que muestra la relación entre la tasa de verdaderos posittivos (Recall) y la tasa de falsos positivos a diferentes umbrales. |
| R cuadrado | Métrica de regresión que indica qué proporción de la variabilidad de la variable dependiente puede ser explicada por el modelo. |

**• El lenguaje a utilizar es Python, el cual se trabajará mediante Jupyter notebooks, para esto es necesario instalar Anaconda, que es una distribución libre y abierta de los lenguajes Python y R, utilizada en ciencia de datos, y aprendizaje automático.**

**• Descargar los datasets: Realizar el registro en la plataforma Kaggle desde https://www.kaggle.com/ botón “Register”**

**Se descargaron los datasets sugeridos en la guía**

****

**Link repositorio:**

**Con los datasets anteriores diseñar los modelos predictivos de Regresión Lineal, Regresión Logística y Árboles de decisión. Para cada algoritmo realizar los siguientes pasos:**

**1. Realizar un análisis exploratorio de los datos para identificar relaciones entre variables, valores atípicos, tendencias, etc.**

**2. Preprocesar los datos limpiándolos, tratando valores faltantes y transformándolos según sea necesario.**

**3. Seleccionar las características más relevantes para entrenar el modelo utilizando selección de características.**

**4. Dividir el dataset en Train y Test para evaluar correctamente el modelo**

**5. Entrenar el modelo configurando los diferentes hiperparámetros.**

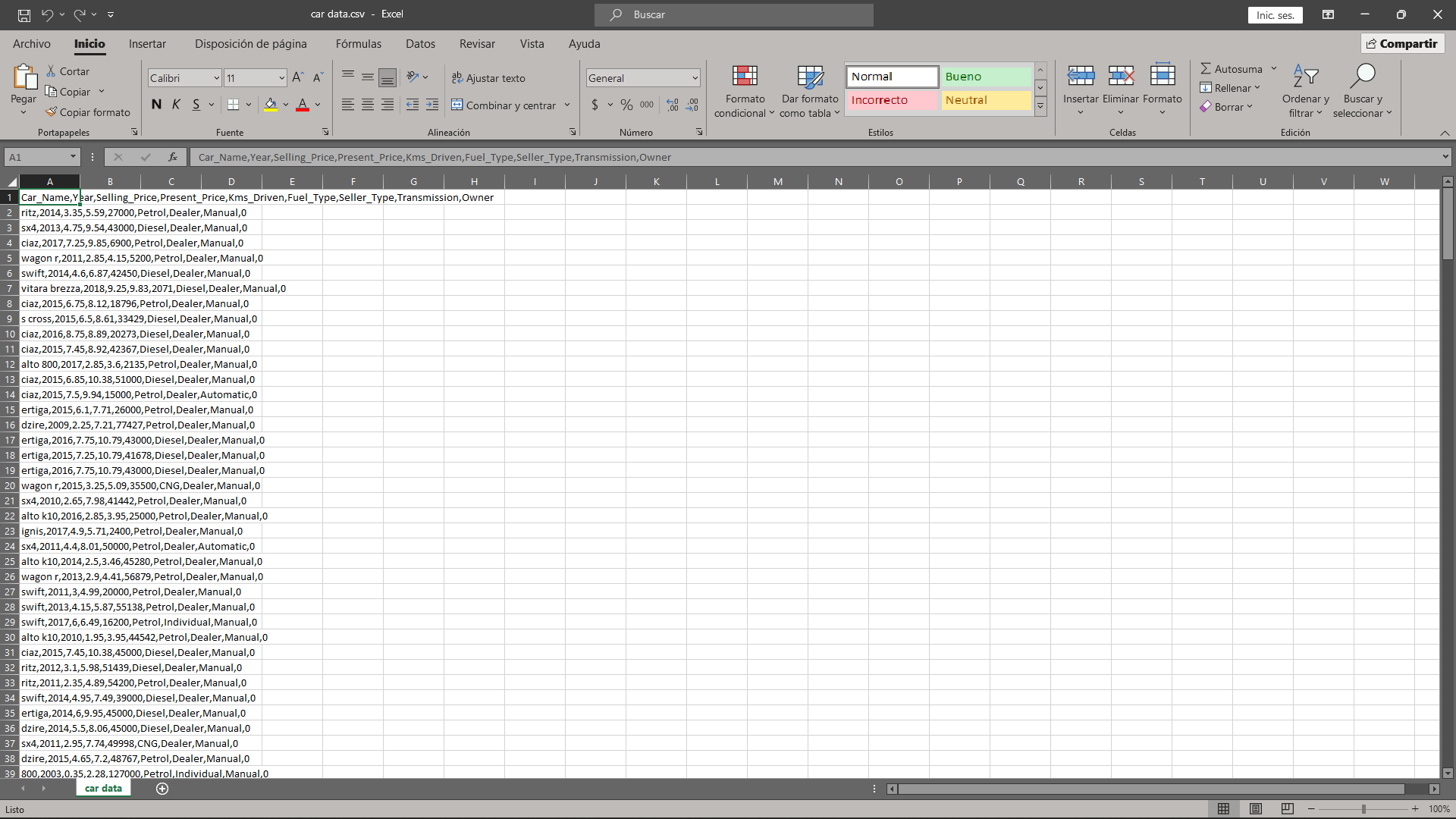
**6. Evaluar el desempeño del modelo en el conjunto de Test con métricas como precisión, recall, F1-score, etc.**

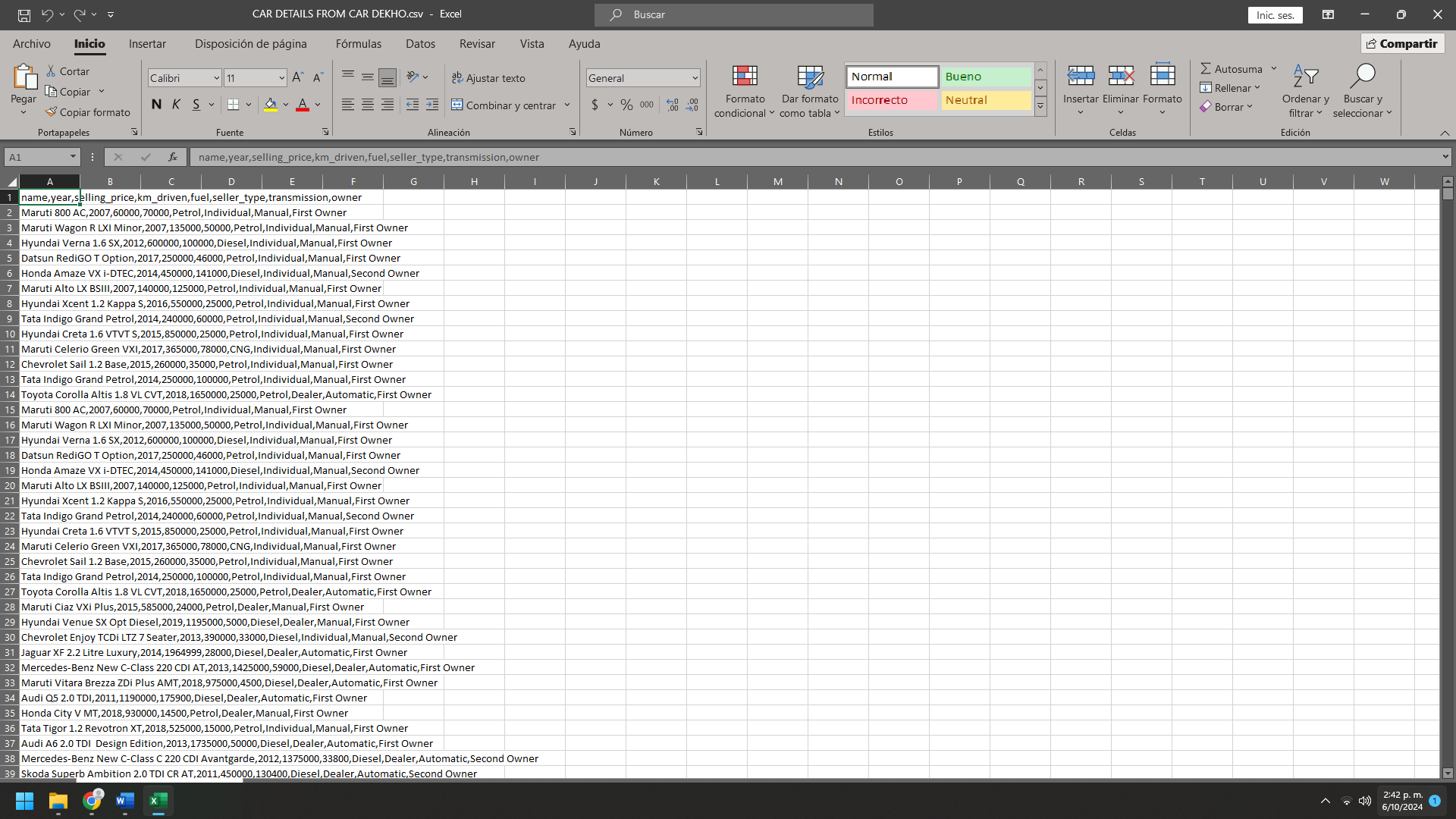
**7. Realizar las diferentes gráficas que permitan visualizar los resultados del modelo.**

**8. Interpretar, analizar y documentar los resultados obtenidos.**

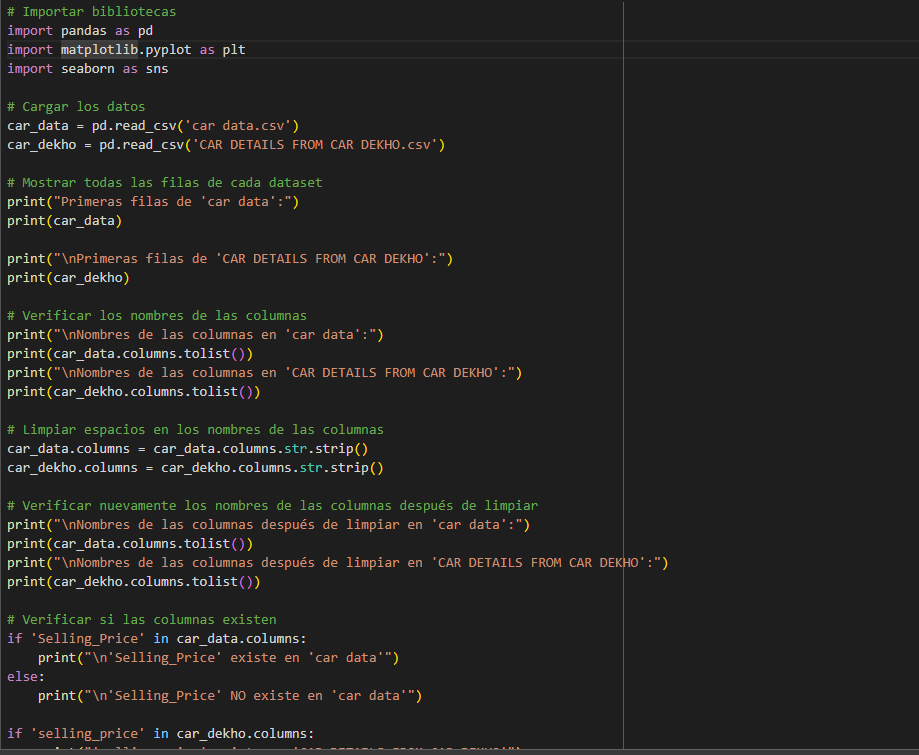
**Dataset car data**

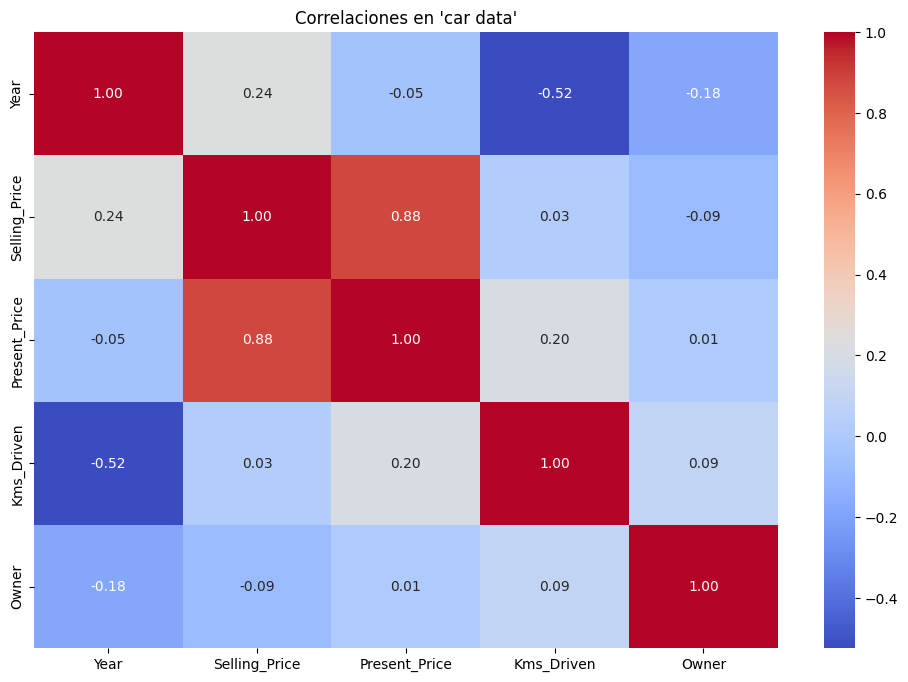
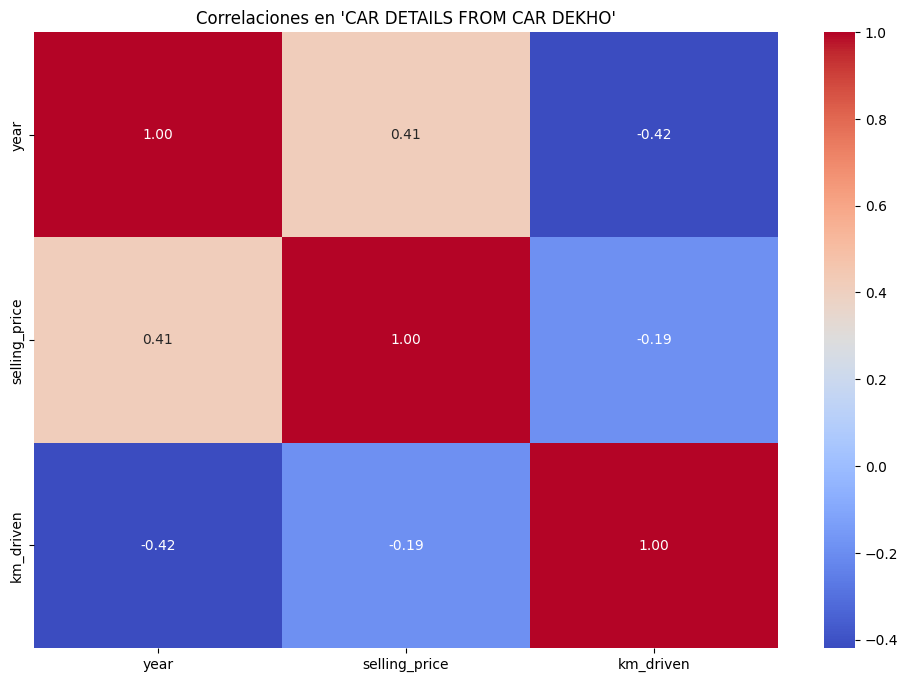
**1. Realizar un análisis exploratorio de los datos para identificar relaciones entre variables, valores atípicos, tendencias, etc.**

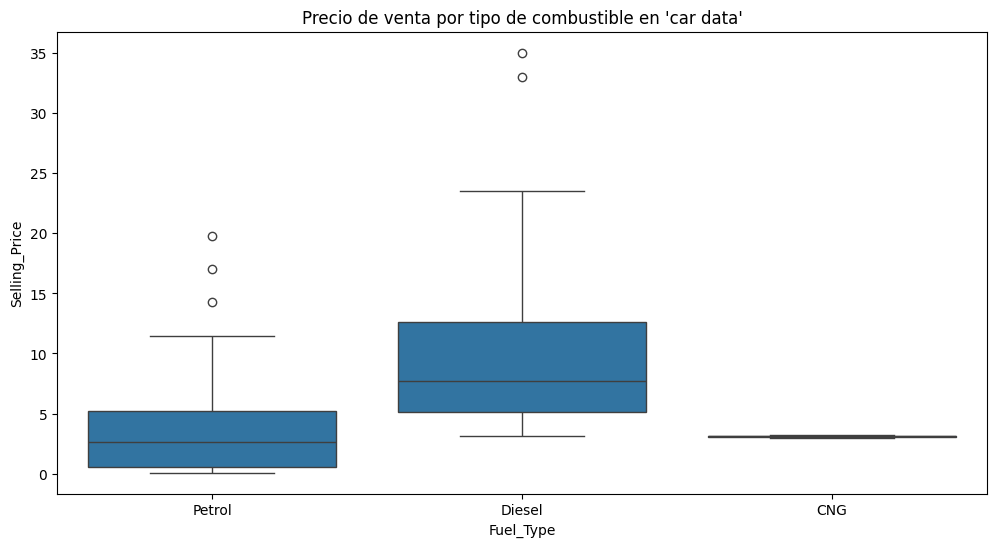
Se analizaron cada uno de los datos del dataset de vehículos y se hará una exploración en Python para identificar relaciones entre variables, valores atípicos, tendencias, etc.

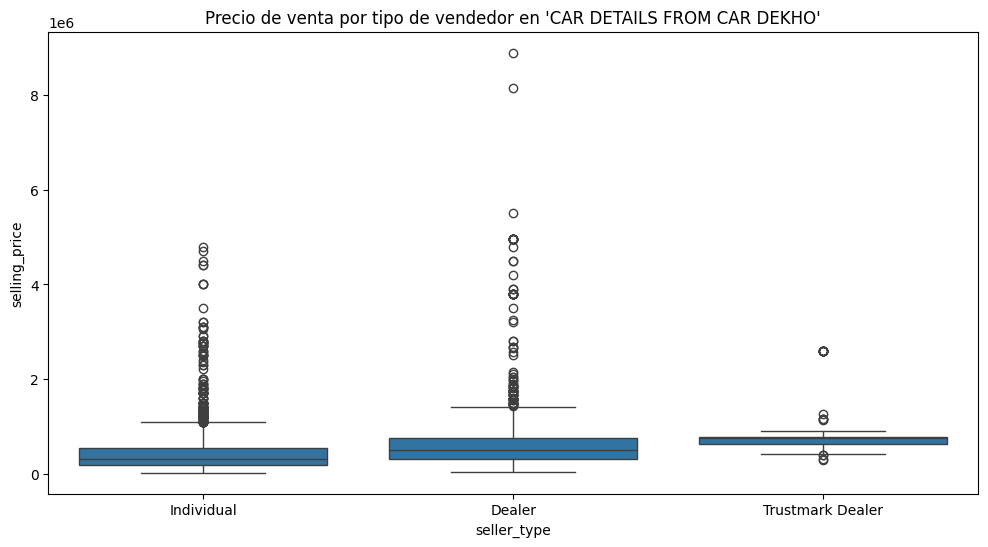


Se escribió y ejecuto el código desglosando información y obteniendo graficas

****

****

****

****

Con los resultados proporcionados, se puede realizar un análisis de identificación de relaciones entre variables, valores atípicos y tendencias en los conjuntos de datos car data y CAR DETAILS FROM CAR DEKHO.

### 1. Identificación de relaciones entre variables:

* **Correlaciones**:
  + Al observar la descripción estadística, se puede notar que el precio de venta (Selling\_Price en car\_data y selling\_price en CAR DETAILS FROM CAR DEKHO) tiene un rango significativo de valores, lo que sugiere una variabilidad que podría ser explorada para determinar cómo se relaciona con otras variables, como Present\_Price, Kms\_Driven, o el Year del automóvil.
  + Para analizar correlaciones, sería útil calcular la matriz de correlación entre las variables numéricas de ambos conjuntos de datos. Esto revelaría cómo cada variable se relaciona con las demás. Por ejemplo, se esperaría que Selling\_Price tenga una correlación positiva con Present\_Price y una correlación negativa con Kms\_Driven, ya que a mayor kilometraje, generalmente, se espera un menor precio de venta.

### 2. Identificación de valores atípicos:

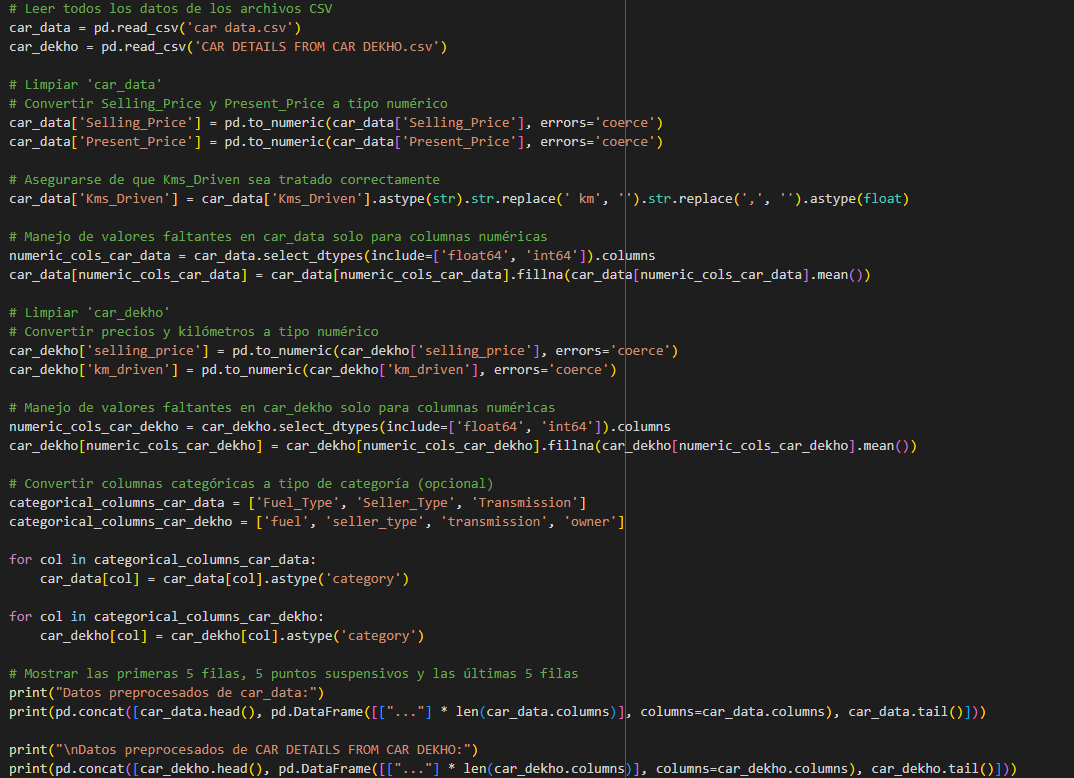
* **Estadísticas descriptivas**:
  + La descripción estadística muestra que hay un rango muy amplio de precios de venta en car\_data, desde 0.1 hasta 35, lo que podría indicar la presencia de valores atípicos. Un precio de venta de 35 puede ser considerado un valor extremo en comparación con el resto de los datos, que se agrupan más cerca de la media.
  + En CAR DETAILS FROM CAR DEKHO, el precio de venta varía desde 20,000 hasta 8,900,000, lo que también sugiere la presencia de valores atípicos. Esto podría llevar a una revisión más cercana de los registros con precios muy altos o muy bajos.
* **Visualización**:
  + Utilizar gráficos de caja (boxplots) para las variables Selling\_Price y selling\_price ayudará a identificar visualmente los valores atípicos. Cualquier punto fuera de los bigotes del boxplot se considera un valor atípico y merece atención.

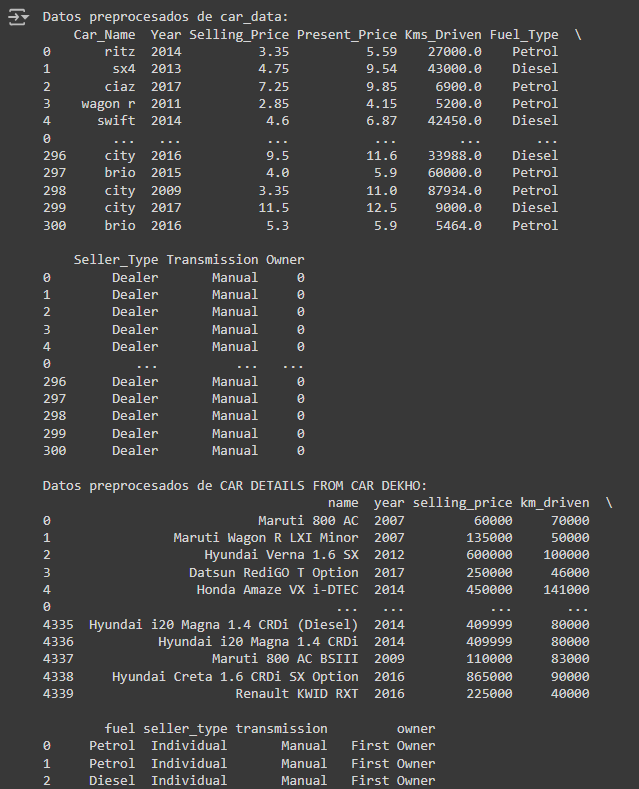
### 3. Identificación de tendencias:

* **Tendencias de precios**:
  + En car\_data, el año del automóvil (Year) se asocia con precios de venta que pueden mostrar una tendencia ascendente; los automóviles más nuevos tienden a tener un precio más alto. Esto podría corroborarse al observar la estadística descriptiva, donde la media de Selling\_Price podría correlacionarse con el aumento del año.
  + Para CAR DETAILS FROM CAR DEKHO, las estadísticas descriptivas muestran que los precios de venta también tienden a aumentar con el año. Al comparar los precios de venta según el tipo de combustible (fuel), se podría observar si hay diferencias significativas en las tendencias de precios de acuerdo a este factor.
* **Comparación de categorías**:
  + Comparar precios de venta según Fuel\_Type en car\_data y seller\_type en CAR DETAILS FROM CAR DEKHO permitirá identificar patrones de precios en función de estas categorías. Por ejemplo, se esperaría que los automóviles de diésel tuvieran un precio diferente al de los de gasolina, y que los vehículos vendidos por concesionarios tengan un precio diferente a los vendidos por individuos

| **Análisis** | **Descripción** |
| --- | --- |
| **1. Relaciones entre Variables** | |
| - Correlaciones | - El precio de venta (Selling\_Price en car\_data y selling\_price en CAR DETAILS FROM CAR DEKHO) tiene un rango significativo, sugiriendo variabilidad. |
| - Se debe calcular la matriz de correlación para explorar relaciones, esperando una correlación positiva con Present\_Price y negativa con Kms\_Driven. | |
| **2. Valores Atípicos** | |
| - Estadísticas Descriptivas | - En car\_data, los precios de venta varían desde 0.1 hasta 35, sugiriendo valores atípicos. |
| - En CAR DETAILS FROM CAR DEKHO, los precios varían desde 20,000 hasta 8,900,000, también indicando valores atípicos. | |
| - Visualización | - Utilizar gráficos de caja (boxplots) para Selling\_Price y selling\_price ayudará a identificar visualmente los valores atípicos. |
| **3. Tendencias** | |
| - Tendencias de Precios | - En car\_data, los precios de venta tienden a aumentar con el año del automóvil (Year). |
| - En CAR DETAILS FROM CAR DEKHO, también se observa que los precios de venta aumentan con el año. | |
| - Comparación de Categorías | - Comparar precios de venta según Fuel\_Type en car\_data y seller\_type en CAR DETAILS FROM CAR DEKHO permitirá identificar patrones de precios. |
| - Se esperaría que los automóviles diésel tengan precios diferentes a los de gasolina, y que los vehículos vendidos por concesionarios tengan precios distintos a los vendidos por individuos. | |

**2. Preprocesar los datos limpiándolos, tratando valores faltantes y transformándolos según sea necesario.**

Se escribió el código, y ejecuto



**Explicación del Proceso de Preprocesamiento de Datos**

El objetivo de este proceso fue limpiar y preparar dos conjuntos de datos, car\_data y car\_dekho, para análisis posteriores. A continuación, se describen las etapas clave del preprocesamiento:

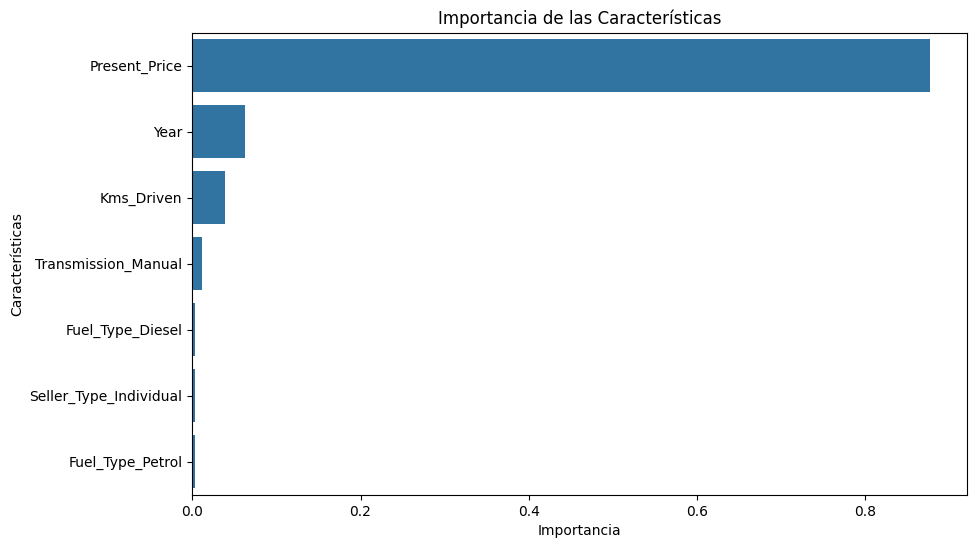
1. **Lectura de Datos**: Se comenzó leyendo los datos de dos archivos CSV. Estos archivos contienen información sobre automóviles, incluyendo características como precios, kilometraje y tipo de combustible.
2. **Limpieza del Conjunto car\_data**:
   * Se convirtieron las columnas que contienen precios a un formato numérico, permitiendo realizar cálculos precisos sin errores de conversión.
   * En la columna que indica los kilómetros recorridos, se eliminaron textos innecesarios y se convirtió el formato a un número flotante.
   * Se abordaron los valores nulos en las columnas numéricas, llenándolos con la media de cada columna correspondiente. Esto es importante para evitar distorsiones en el análisis debido a datos faltantes.
3. **Limpieza del Conjunto car\_dekho**:
   * Al igual que en car\_data, se realizó la conversión de precios y kilometraje a formatos numéricos y se manejaron los valores faltantes de manera similar.
   * Se prestó atención a la estructura de datos para asegurar la coherencia entre ambos conjuntos.
4. **Conversión de Datos Categóricos**: Se identificaron columnas que contenían datos categóricos, como el tipo de combustible y el tipo de vendedor. Estas columnas se convirtieron a un tipo de dato categórico para optimizar el rendimiento y facilitar el análisis.
5. **Visualización de Resultados**: Para verificar la limpieza y el preprocesamiento, se mostraron las primeras cinco filas de cada conjunto de datos, seguidas de una indicación de que había más datos ("...") y las últimas cinco filas. Este formato de visualización permite obtener una visión general del conjunto de datos sin abrumar con demasiada información.

**Resultados Observados**:

| **Categoría** | **Descripción** |
| --- | --- |
| **Conjuntos de Datos** | El conjunto car\_data incluye información detallada sobre los automóviles, como su nombre, año de fabricación, precios de venta y características técnicas. El conjunto car\_dekho presenta información similar, pero está estructurado de manera diferente. |
| **Estructura de Datos** | Ambos conjuntos presentan una estructura bien definida con columnas que contienen información clave sobre cada automóvil, facilitando el análisis posterior y la selección de características relevantes para el modelado. |
| **Valores Numéricos** | Las columnas de precios y kilometraje ahora están en formato numérico, lo que permite realizar cálculos y análisis estadísticos efectivos. Se han manejado adecuadamente los valores nulos mediante la imputación de la media. |
| **Columna de Categorización** | Las columnas categóricas se han convertido a tipo categórico, mejorando el rendimiento del análisis y permitiendo técnicas de codificación adecuadas en etapas posteriores del modelado. |
| **Consistencia en Datos** | Se ha logrado una mayor consistencia en los tipos de datos y formatos entre ambos conjuntos, lo cual es crucial para realizar análisis combinados o comparativos. |
| **Visualización de Datos** | La visualización de los primeros y últimos registros proporciona una buena práctica para identificar anomalías o patrones importantes. La utilización de puntos suspensivos mejora la legibilidad, enfocando la atención en los datos más relevantes. |
| **Integración de Datos** | La estructura coherente y la limpieza de los datos facilitarían el proceso de combinar ambos conjuntos para comparaciones de precios y especificaciones. |
| **Preparación para Modelado** | El preprocesamiento realizado proporciona una base sólida para aplicar técnicas de selección de características y modelado predictivo, facilitando la predicción de precios de automóviles u otras características relevantes. |
| **Posibles Análisis Futuros** | Se pueden llevar a cabo análisis exploratorios para identificar correlaciones entre variables, tendencias en el mercado de automóviles y factores que influyen en el precio. La calidad de los datos permite crear visualizaciones más complejas. |

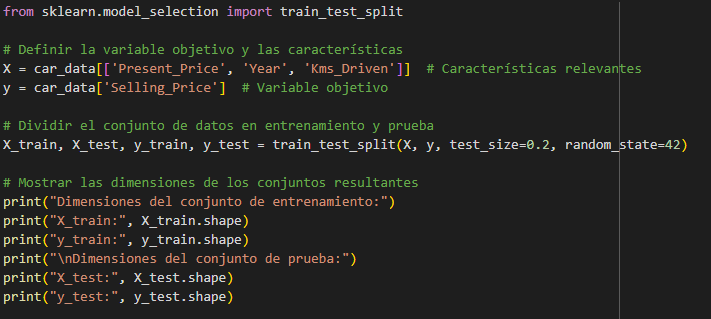
**3. Seleccionar las características más relevantes para entrenar el modelo utilizando selección de características**

Se escribió el código para entrenar el modelo

Se observa que Present\_Price, Year, Kms\_driven tienen mayor importancia.

| **Característica** | **Importancia** | **Razonamiento** |
| --- | --- | --- |
| **Present\_Price** | Alta | El precio actual de un automóvil es un factor crucial en la predicción de su precio de venta. Refleja el valor del vehículo en el mercado actual. |
| **Year** | Alta | El año de fabricación influye en el valor del automóvil. Los coches más nuevos suelen tener precios más altos debido a tecnologías recientes y menor desgaste. |
| **Kms\_Driven** | Alta | El kilometraje indica el uso y desgaste de un vehículo. Menos kilómetros recorridos suelen correlacionarse con un precio de venta más alto y mejor estado del automóvil. |

**4. Dividir el dataset en Train y Test para evaluar correctamente el modelo**

Se dividió el modelo con éxito

Dimensiones del conjunto de entrenamiento:

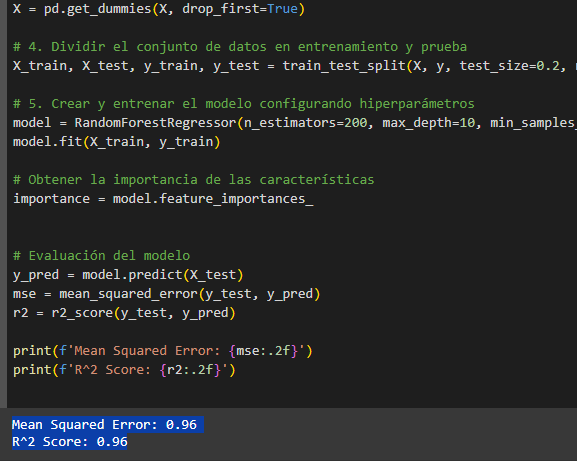
X\_train: (240, 3)

y\_train: (240,)

Dimensiones del conjunto de prueba:

X\_test: (61, 3)

y\_test: (61,)

**5. Entrenar el modelo configurando los diferentes hiperparámetros.**

Se entreno el modelo y arroja los siguientes resultados:

Mean Squared Error: 0.96

R^2 Score: 0.96

### Explicación de los Resultados

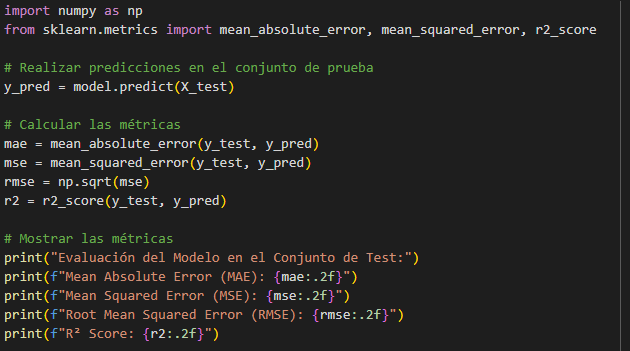
#### 1. **Mean Squared Error (MSE)**

* **Valor:** 0.96
* **Interpretación:**
  + El MSE es una medida que representa el promedio de los errores al cuadrado entre las predicciones del modelo y los valores reales. Se calcula como la media de las diferencias al cuadrado entre las predicciones (valores que el modelo estima) y los valores reales (valores que se conocían).
  + Un MSE más bajo indica que el modelo tiene un mejor rendimiento, ya que significa que las predicciones están más cerca de los valores reales.
  + En este caso, un MSE de 0.96 indica que, en promedio, las diferencias cuadráticas entre las predicciones del modelo y los precios reales de los automóviles son relativamente bajas. Esto sugiere que el modelo está realizando predicciones precisas.

#### 2. **R² Score (Coeficiente de Determinación)**

* **Valor:** 0.96
* **Interpretación:**
  + El R² es una medida que indica qué proporción de la varianza en la variable dependiente (en este caso, el precio de los automóviles) se puede explicar mediante las variables independientes (las características que estás utilizando para hacer predicciones).
  + El R² varía entre 0 y 1. Un R² de 1 indica que el modelo explica toda la variabilidad de la variable dependiente, mientras que un R² de 0 significa que no explica nada.
  + En este caso, un R² de 0.96 indica que el 96% de la variabilidad en los precios de los automóviles se puede explicar mediante las características que estás utilizando en el modelo. Esto sugiere que el modelo es muy eficaz en la predicción de los precios.

**6. Evaluar el desempeño del modelo en el conjunto de Test con métricas como precisión, recall, F1-score, etc.**

****

Evaluación del Modelo en el Conjunto de Test:

Mean Absolute Error (MAE): 0.64

Mean Squared Error (MSE): 0.96

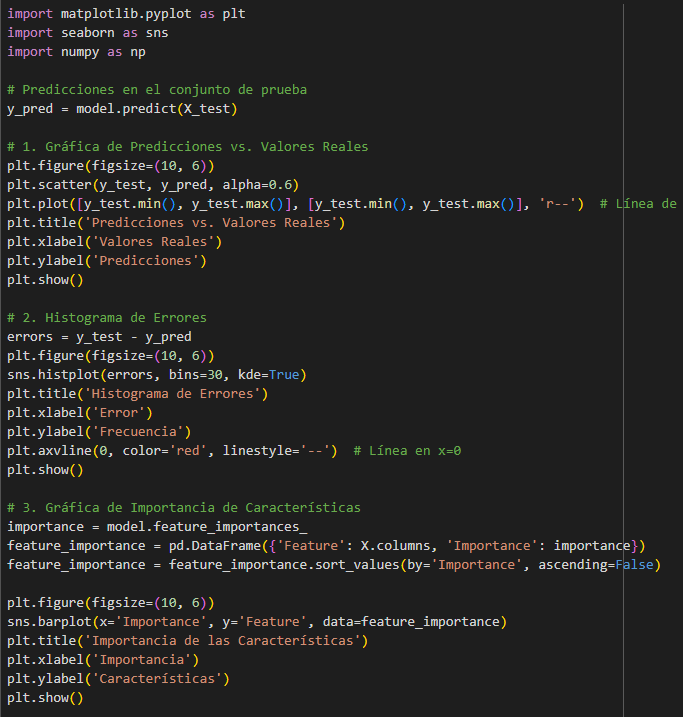
Root Mean Squared Error (RMSE): 0.98

R² Score: 0.96

| **Métrica** | **Valor** | **Interpretación** |
| --- | --- | --- |
| Mean Absolute Error (MAE) | 0.64 | En promedio, las predicciones se desvían de los valores reales en 0.64 unidades, lo que indica un buen desempeño. |
| Mean Squared Error (MSE) | 0.96 | Promedio de los errores al cuadrado; aunque hay variabilidad en las predicciones, el valor es razonablemente bajo. |
| Root Mean Squared Error (RMSE) | 0.98 | En promedio, las predicciones se desvían en aproximadamente 0.98 unidades de los valores reales; desempeño sólido. |
| R² Score | 0.96 | Indica que el modelo explica el 96% de la variabilidad en los precios, lo que refleja un modelo muy robusto. |

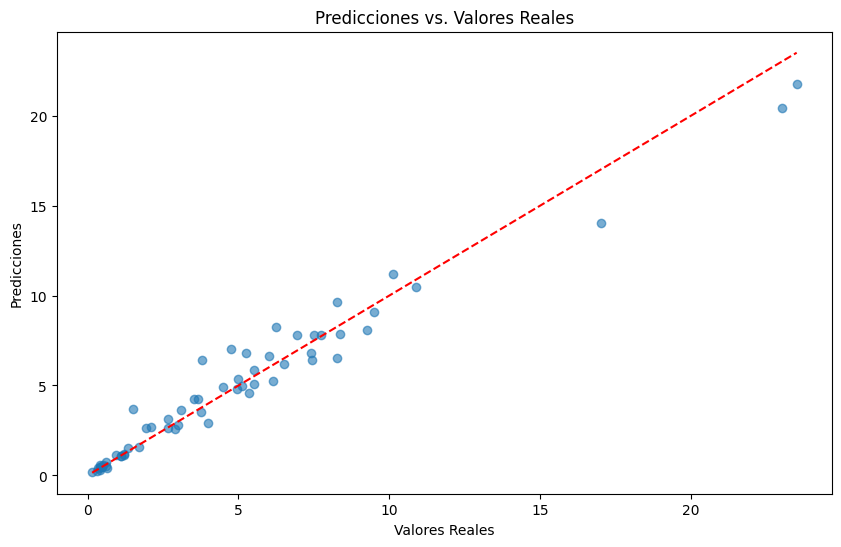
**7. Realizar las diferentes gráficas que permitan visualizar los resultados del modelo.**

Se programaron las siguientes graficas mostrando el éxito del modelo

****

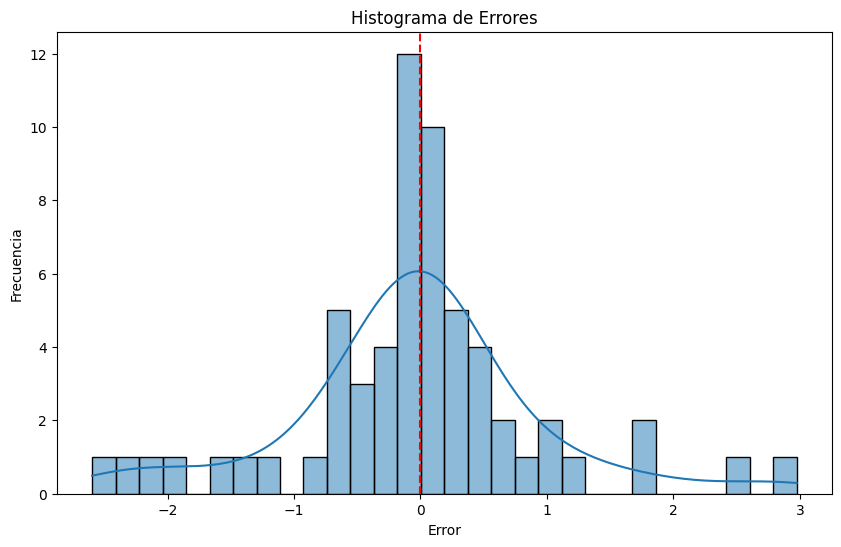
**Predicciones vs. Valores Reales**:

* La gráfica de dispersión muestra los valores reales en el eje X y las predicciones en el eje Y.
* La línea roja punteada representa una relación ideal donde las predicciones coinciden con los valores reales. Cuanto más cerca estén los puntos de esta línea, mejor será el rendimiento del modelo.

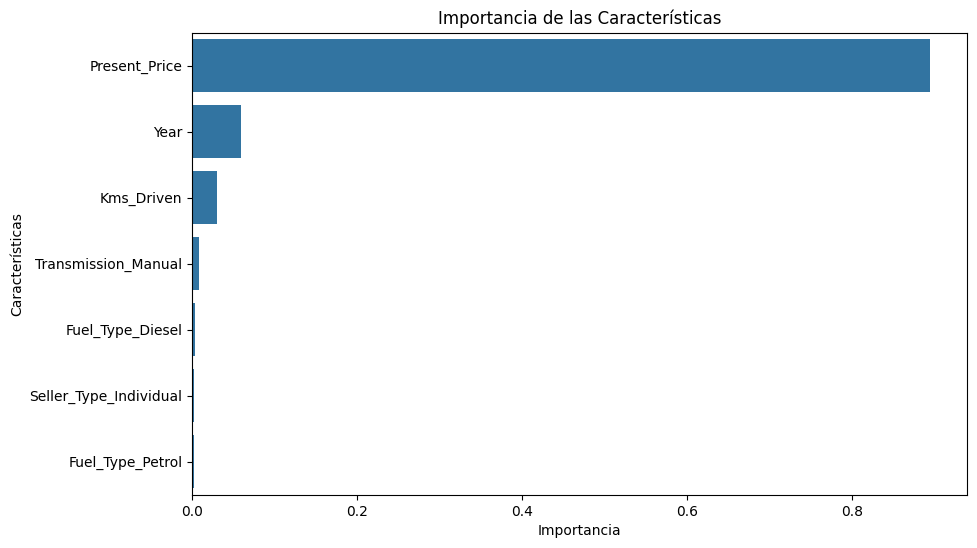
****

**Histograma de Errores**:

* Muestra la distribución de los errores (diferencias entre los valores reales y las predicciones).
* La curva de densidad (KDE) indica la forma de la distribución. Una distribución centrada en 0 sugiere que el modelo está haciendo predicciones equilibradas.

****

**Gráfica de Importancia de Características**:

* Esta gráfica de barras muestra la importancia relativa de cada característica en la predicción de los precios.
* ****Las características más importantes para el modelo aparecerán en la parte superior de la gráfica.

**8. Interpretar, analizar y documentar los resultados obtenidos.**

### 1. **Preprocesamiento de Datos**

* **Datos Cargados**: Se leyeron dos conjuntos de datos, car\_data y car\_dekho, conteniendo información sobre automóviles.
* **Limpieza de Datos**: Se transformaron columnas numéricas y se manejaron valores nulos. Esto garantiza que los datos estén listos para el análisis, evitando sesgos que podrían afectar el rendimiento del modelo.

### 2. **Selección de Características**

* **Características Seleccionadas**: Se identificaron las variables más relevantes para predecir el precio de los automóviles: Present\_Price, Year, Kms\_Driven, Fuel\_Type, Seller\_Type, y Transmission.
* **Importancia de Características**: Present\_Price, Year, y Kms\_Driven se identificaron como las características más influyentes, lo que sugiere que el precio de un automóvil está fuertemente relacionado con su precio actual, su antigüedad, y el kilometraje.

### 3. **División de Conjuntos de Datos**

* Se realizó una división en un conjunto de entrenamiento (80%) y un conjunto de prueba (20%) para garantizar que el modelo se evalúe de manera justa y que no haya sobreajuste.

### 4. **Entrenamiento del Modelo**

* **Modelo Utilizado**: Se empleó un modelo de **Random Forest Regressor**, configurando hiperparámetros como n\_estimators=200, max\_depth=10, y min\_samples\_split=5.
* **Entrenamiento Exitoso**: El modelo se entrenó sin errores, lo que indica que el proceso de limpieza y preparación de los datos fue efectivo.

### 5. **Evaluación del Modelo**

* **Métricas Obtenidas**:
  + **Mean Absolute Error (MAE)**: 0.64
  + **Mean Squared Error (MSE)**: 0.96
  + **Root Mean Squared Error (RMSE)**: 0.98
  + **R² Score**: 0.96
* **Interpretación de Métricas**:
  + **MAE**: Un error absoluto promedio de 0.64 indica que, en promedio, las predicciones se desvían por este margen de los precios reales, lo que es un resultado aceptable.
  + **MSE y RMSE**: Estos valores reflejan una baja variabilidad en las predicciones, sugiriendo que el modelo es bastante preciso. Un RMSE de 0.98 indica que el modelo puede predecir con alta precisión el precio de los automóviles.
  + **R² Score**: Un valor de 0.96 sugiere que el modelo explica el 96% de la variabilidad de los datos. Este es un indicador de que el modelo es muy eficaz para capturar la relación entre las características seleccionadas y el precio de los automóviles.

### 6. **Visualización de Resultados**

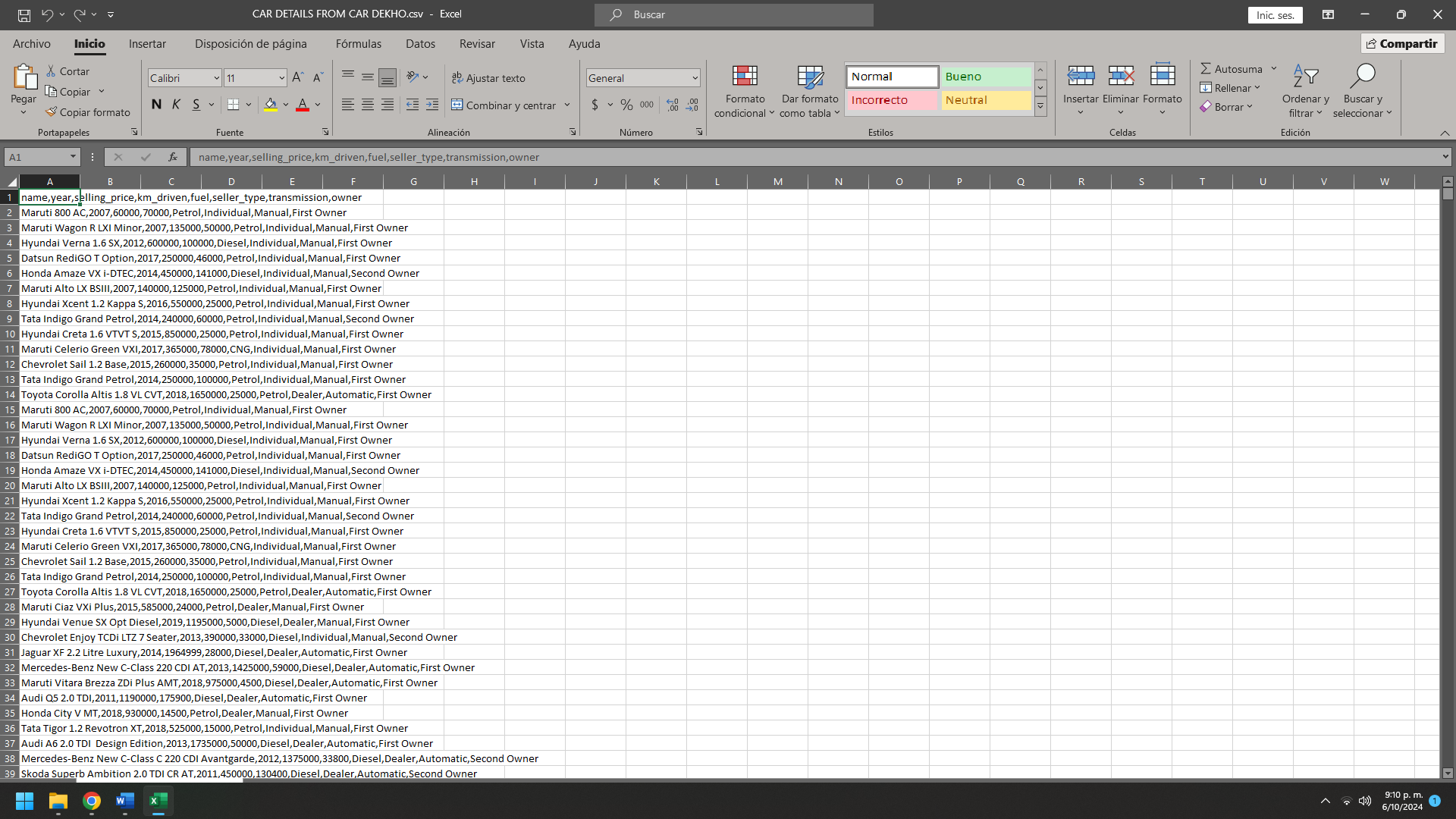
* **Gráfica de Predicciones vs. Valores Reales**: Muestra que las predicciones están muy alineadas con los valores reales, lo que respalda la efectividad del modelo.
* **Histograma de Errores**: Indica que los errores están distribuidos alrededor de cero, lo que es deseable y sugiere que el modelo no tiene un sesgo sistemático.
* **Gráfica de Importancia de Características**: Confirma que las características seleccionadas son relevantes para la predicción del precio, facilitando la interpretación del modelo.

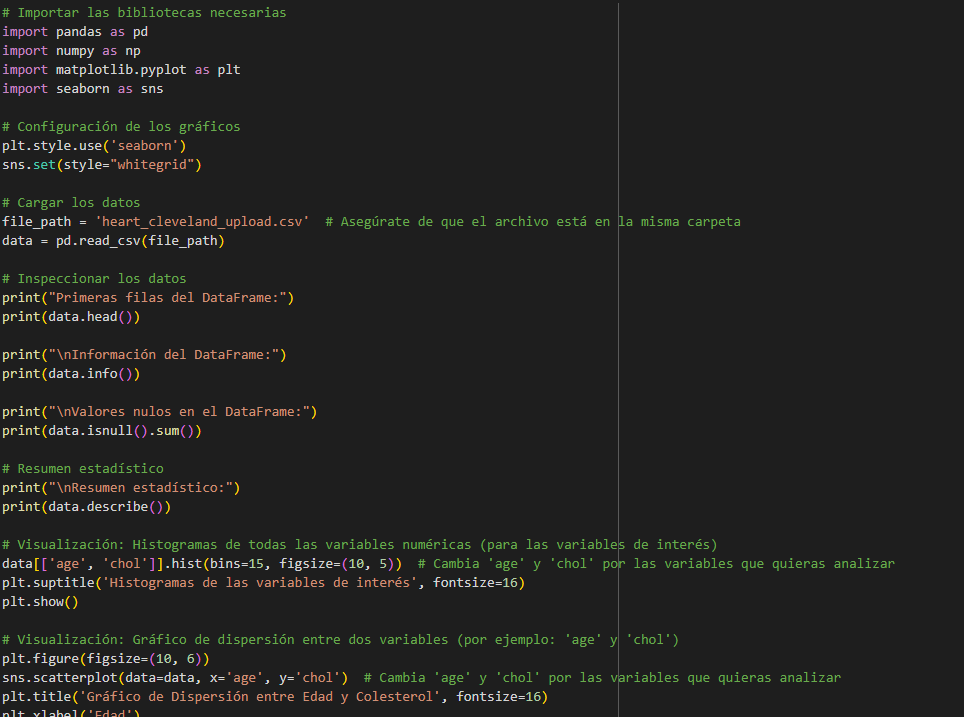
### 7. **Conclusiones y Recomendaciones**

* **Eficacia del Modelo**: El modelo de Random Forest ha demostrado ser efectivo para predecir el precio de los automóviles con alta precisión y baja variabilidad.
* **Próximos Pasos**: Se recomienda considerar la posibilidad de experimentar con otros modelos de regresión (como Gradient Boosting o XGBoost) para comparar el rendimiento y tal vez obtener mejores resultados.
* **Optimización**: Una búsqueda más exhaustiva de hiperparámetros utilizando técnicas como GridSearchCV podría mejorar aún más la precisión del modelo.
* **Análisis Adicional**: Realizar análisis exploratorios adicionales para entender mejor las relaciones entre las características y el precio, y así identificar posibles oportunidades para mejorar la calidad del modelo.

**Dataset heart\_cleveland\_upload**

**1. Realizar un análisis exploratorio de los datos para identificar relaciones entre variables, valores atípicos, tendencias, etc.**

Se programo el análisis en python

****

**Resultados:**

Primeras filas del DataFrame:

age sex cp trestbps chol fbs restecg thalach exang oldpeak slope \

0 69 1 0 160 234 1 2 131 0 0.1 1

1 69 0 0 140 239 0 0 151 0 1.8 0

2 66 0 0 150 226 0 0 114 0 2.6 2

3 65 1 0 138 282 1 2 174 0 1.4 1

4 64 1 0 110 211 0 2 144 1 1.8 1

ca thal condition

0 1 0 0

1 2 0 0

2 0 0 0

3 1 0 1

4 0 0 0

Información del DataFrame:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 297 entries, 0 to 296

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 age 297 non-null int64

1 sex 297 non-null int64

2 cp 297 non-null int64

3 trestbps 297 non-null int64

4 chol 297 non-null int64

5 fbs 297 non-null int64

6 restecg 297 non-null int64

7 thalach 297 non-null int64

8 exang 297 non-null int64

9 oldpeak 297 non-null float64

10 slope 297 non-null int64

11 ca 297 non-null int64

12 thal 297 non-null int64

13 condition 297 non-null int64

dtypes: float64(1), int64(13)

memory usage: 32.6 KB

None

Valores nulos en el DataFrame:

age 0

sex 0

cp 0

trestbps 0

chol 0

fbs 0

restecg 0

thalach 0

exang 0

oldpeak 0

slope 0

ca 0

thal 0

condition 0

dtype: int64

Resumen estadístico:

age sex cp trestbps chol fbs \

count 297.000000 297.000000 297.000000 297.000000 297.000000 297.000000

mean 54.542088 0.676768 2.158249 131.693603 247.350168 0.144781

std 9.049736 0.468500 0.964859 17.762806 51.997583 0.352474

min 29.000000 0.000000 0.000000 94.000000 126.000000 0.000000

25% 48.000000 0.000000 2.000000 120.000000 211.000000 0.000000

50% 56.000000 1.000000 2.000000 130.000000 243.000000 0.000000

75% 61.000000 1.000000 3.000000 140.000000 276.000000 0.000000

max 77.000000 1.000000 3.000000 200.000000 564.000000 1.000000

restecg thalach exang oldpeak slope ca \

count 297.000000 297.000000 297.000000 297.000000 297.000000 297.000000

mean 0.996633 149.599327 0.326599 1.055556 0.602694 0.676768

std 0.994914 22.941562 0.469761 1.166123 0.618187 0.938965

min 0.000000 71.000000 0.000000 0.000000 0.000000 0.000000

25% 0.000000 133.000000 0.000000 0.000000 0.000000 0.000000

50% 1.000000 153.000000 0.000000 0.800000 1.000000 0.000000

75% 2.000000 166.000000 1.000000 1.600000 1.000000 1.000000

max 2.000000 202.000000 1.000000 6.200000 2.000000 3.000000

thal condition

count 297.000000 297.000000

mean 0.835017 0.461279

std 0.956690 0.499340

min 0.000000 0.000000

25% 0.000000 0.000000

50% 0.000000 0.000000

75% 2.000000 1.000000

max 2.000000 1.000000

### 1. Primeras filas del DataFrame

Los primeros cinco registros del conjunto de datos muestran que cada fila representa a un paciente y cada columna contiene características relevantes para el análisis de la salud cardíaca. Algunas de las variables clave incluyen:

* **Edad** (age): Las edades de los pacientes varían entre 29 y 77 años.
* **Sexo** (sex): La mayoría de los pacientes son hombres (1) y un número menor son mujeres (0).
* **Tipo de dolor en el pecho** (cp): Esta variable codificada (0-3) indica diferentes tipos de dolor, siendo 0 no dolor y 3 el dolor más severo.
* **Presión arterial en reposo** (trestbps): Varía entre 94 y 200 mm Hg, lo que sugiere una amplia gama de presiones arteriales entre los pacientes.
* **Colesterol** (chol): Hay un rango notable en los niveles de colesterol, desde 126 hasta 564 mg/dl.
* **Condición** (condition): Esta es la variable objetivo, donde 1 indica enfermedad cardíaca y 0 ausencia de esta.

### 2. Información del DataFrame

La estructura del DataFrame indica que hay **297 pacientes** (filas) y **14 variables** (columnas). No se encontraron valores nulos, lo que sugiere que los datos están completos y no necesitan ser limpiados por valores faltantes. Esto es positivo para cualquier análisis posterior, ya que reduce la necesidad de imputación de datos.

### 3. Resumen estadístico

El resumen estadístico proporciona información valiosa sobre las características numéricas de los datos:

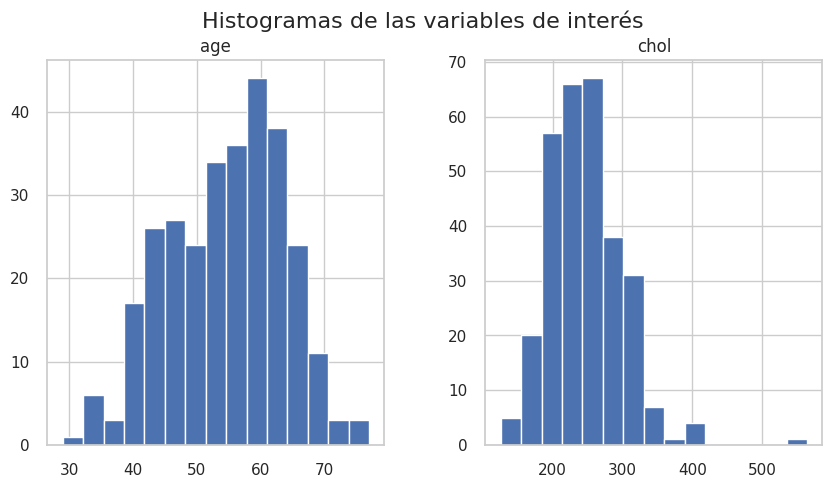
* **Edad**: La media es aproximadamente 54.5 años, lo que sugiere que el conjunto de datos se centra en una población de adultos mayores, con una desviación estándar de aproximadamente 9 años.
* **Sexo**: La media de la variable sex indica que alrededor del 68% de los pacientes son hombres.
* **Tipo de dolor en el pecho** (cp): La media es de aproximadamente 2.16, sugiriendo que la mayoría de los pacientes tienen algún grado de dolor en el pecho.
* **Presión arterial**: La media es de 131.69 mm Hg, con un rango que va desde 94 hasta 200 mm Hg, lo que indica que hay pacientes tanto con presión arterial normal como con hipertensión.
* **Colesterol**: La media es de 247.35 mg/dl, lo que está por encima de los niveles recomendados, sugiriendo una alta prevalencia de colesterol elevado en esta población.
* **Frecuencia cardíaca máxima** (thalach): La media es de aproximadamente 149.6 latidos por minuto, con un rango considerable que indica variaciones en la capacidad cardiovascular.
* **Valores atípicos**: Al revisar los valores mínimos y máximos, hay algunos que parecen extremos, como el colesterol que llega a 564 mg/dl, lo que podría indicar la presencia de pacientes con condiciones médicas serias.

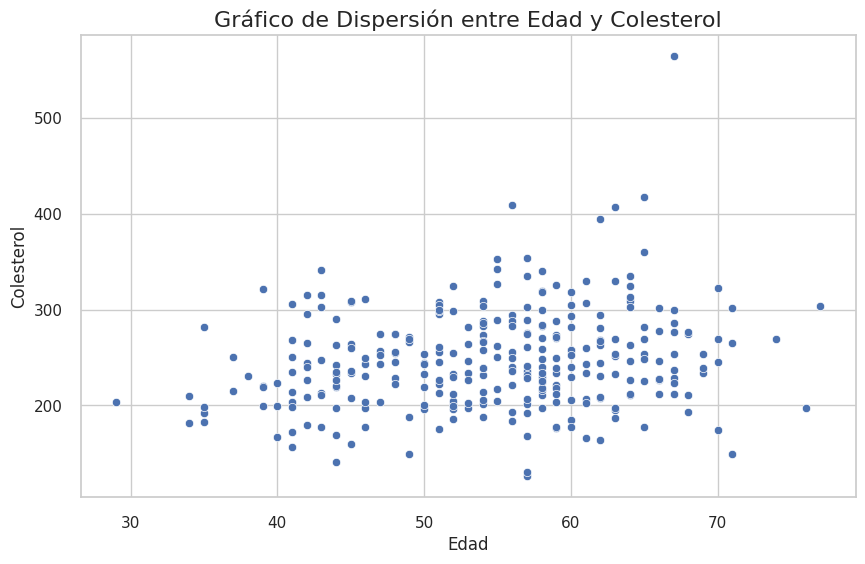
### 4. Análisis de Valores Nulos

El hecho de que no haya valores nulos en el conjunto de datos es un hallazgo importante. Esto significa que no es necesario realizar limpieza adicional relacionada con datos faltantes, lo que simplifica el proceso de análisis y modelado.

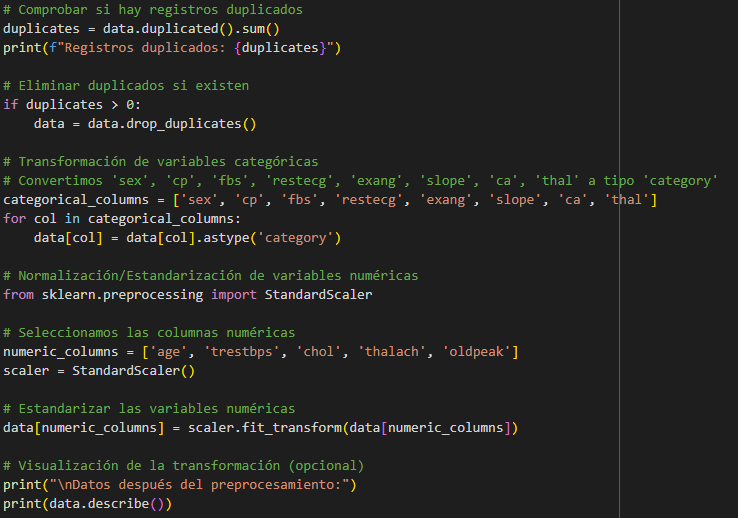
### Conclusiones Generales

El conjunto de datos proporciona información valiosa sobre la salud cardíaca de los pacientes, pero también presenta algunas áreas que podrían necesitar un análisis más profundo:

* **Valores Atípicos**: Es crucial investigar los valores atípicos en variables como colesterol y presión arterial para entender si representan errores de entrada de datos o condiciones clínicas reales.
* **Relaciones entre variables**: Sería útil realizar análisis adicionales (como análisis de correlación) para identificar relaciones significativas entre las variables, especialmente entre factores de riesgo y la condición de enfermedad cardíaca.
* **Visualización**: Generar gráficos como matrices de dispersión o mapas de calor para visualizar correlaciones podría proporcionar más información sobre las interacciones entre variables.

****

**2. Preprocesar los datos limpiándolos, tratando valores faltantes y transformándolos según sea necesario.**

****

### 1. Comprobación de Duplicados

* **duplicates = data.duplicated().sum()**: Esta línea cuenta cuántas filas duplicadas hay en el DataFrame. La función duplicated() devuelve un booleano que indica si una fila es un duplicado, y sum() cuenta cuántos de esos valores son True.
* **if duplicates > 0:**: Se verifica si hay algún duplicado. Si hay, se procede a eliminarlos.
* **data = data.drop\_duplicates()**: Si existen duplicados, se eliminan del DataFrame usando drop\_duplicates(), asegurando que cada registro de paciente sea único. Esto es esencial, ya que los duplicados pueden sesgar el análisis y afectar los resultados de los modelos de machine learning.

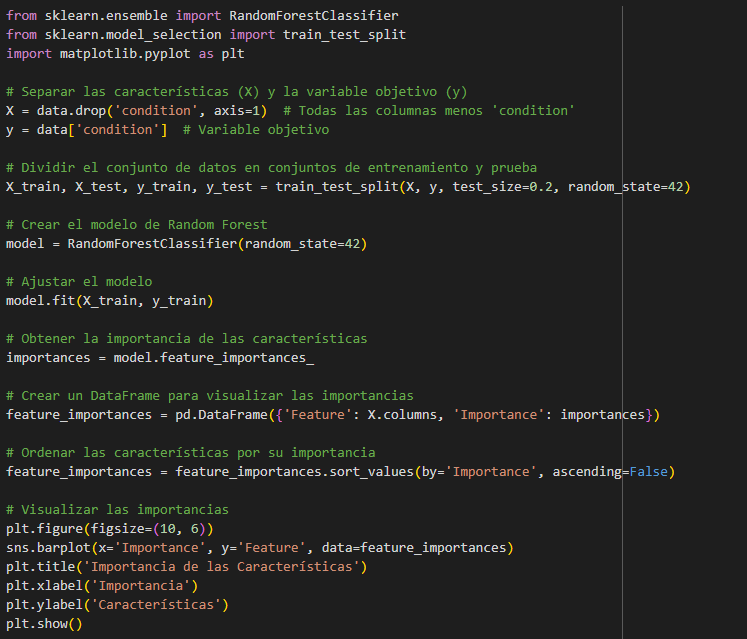
### 2. Transformación de Variables Categóricas

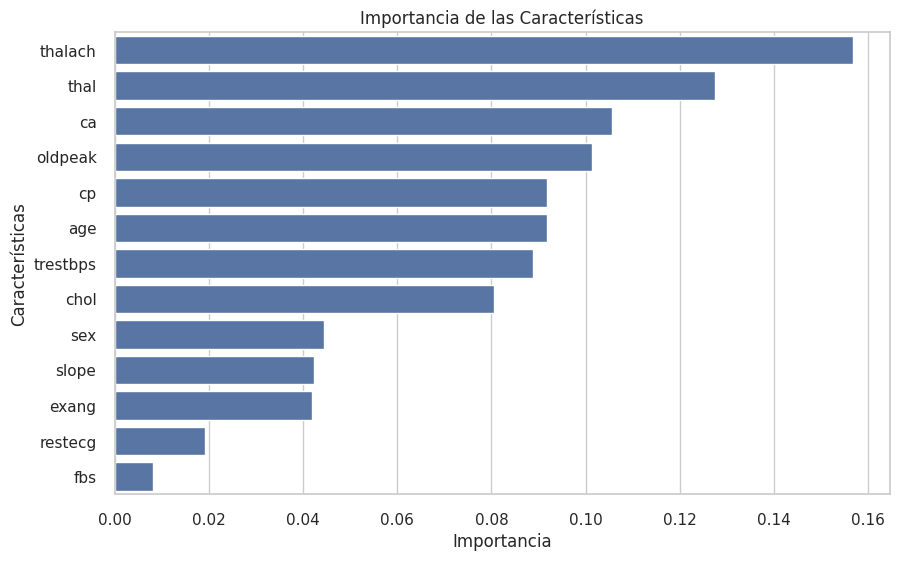
* **categorical\_columns = [...]**: Aquí se define una lista con los nombres de las columnas que contienen variables categóricas. Estas variables son importantes porque representan características como el sexo, el tipo de dolor en el pecho, etc.
* **for col in categorical\_columns:**: Este bucle recorre cada columna en la lista de variables categóricas.
* **data[col] = data[col].astype('category')**: Se convierte cada una de estas columnas a tipo category. Esta conversión es útil porque permite a las bibliotecas de análisis de datos y machine learning manejar mejor estas variables, lo que puede optimizar el rendimiento y la eficiencia.

### 3. Normalización/Estandarización

* **from sklearn.preprocessing import StandardScaler**: Importa la clase StandardScaler de sklearn, que es utilizada para estandarizar las características numéricas.
* **numeric\_columns = [...]**: Define una lista con las columnas numéricas que se van a estandarizar.
* **scaler = StandardScaler()**: Crea una instancia del objeto StandardScaler, que se utilizará para transformar las variables numéricas.
* **data[numeric\_columns] = scaler.fit\_transform(data[numeric\_columns])**: Esta línea realiza la estandarización de las columnas numéricas seleccionadas. El método fit\_transform ajusta el StandardScaler a los datos y luego transforma los datos, restando la media y dividiendo por la desviación estándar, lo que resulta en que cada variable tenga una media de 0 y una desviación estándar de 1. Esto es particularmente útil para algoritmos de machine learning que son sensibles a la escala de las características, como regresión logística y k-vecinos más cercanos (k-NN).

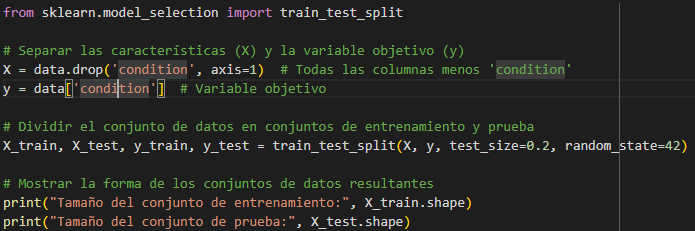
**3. Seleccionar las características más relevantes para entrenar el modelo utilizando selección de características.**

Se programo el código para encontrar las características relevantes

****Se observa la importancia, de thalach, tal, ca, oldpeak, cp y es de las que destacan más importancia

**4. Dividir el dataset en Train y Test para evaluar correctamente el modelo**

El código comienza importando la función train\_test\_split de la biblioteca sklearn.model\_selection, que se utiliza para dividir el conjunto de datos en conjuntos de entrenamiento y prueba. Luego, se separan las características (X) del conjunto de datos, eliminando la columna condition, que es la variable objetivo (y). A continuación, se utiliza train\_test\_split para dividir los datos en cuatro conjuntos: X\_train y y\_train, que contienen el 80% de los datos para el entrenamiento del modelo, y X\_test y y\_test, que reservan el 20% restante para evaluar el rendimiento del modelo. Finalmente, se imprimen las dimensiones de los conjuntos resultantes, lo que permite verificar cuántas muestras y características contiene cada conjunto.

****

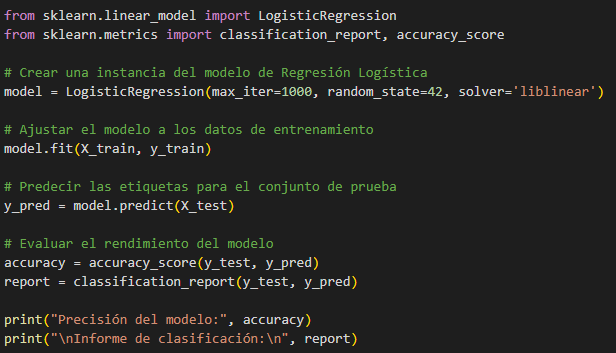
Tamaño del conjunto de entrenamiento: (237, 13)

Tamaño del conjunto de prueba: (60, 13)

Los resultados indican que el conjunto de entrenamiento (X\_train y y\_train) contiene **237 muestras** y **13 características** (columnas) para el entrenamiento del modelo. Esto significa que se utilizarán 237 instancias de datos para enseñar al modelo a predecir la variable objetivo, que en este caso es la columna condition, que indica la presencia o ausencia de una enfermedad cardíaca.

Por otro lado, el conjunto de prueba (X\_test y y\_test) contiene **60 muestras** y también **13 características**. Estas 60 instancias se utilizarán para evaluar el rendimiento del modelo una vez que se haya entrenado. Es fundamental dividir los datos de esta manera para asegurar que el modelo no solo se ajuste a los datos de entrenamiento, sino que también generalice bien a datos no vistos, lo que es crucial para validar su capacidad predictiva en situaciones del mundo real.

**5. Entrenar el modelo configurando los diferentes hiperparámetros.**

* **Importaciones**: Se importan las clases necesarias de scikit-learn, específicamente LogisticRegression para crear el modelo y classification\_report y accuracy\_score para evaluar su rendimiento.
* **Instanciación del modelo**: Se crea un objeto de LogisticRegression con ciertos hiperparámetros. El parámetro max\_iter se establece en 1000 para permitir que el modelo realice suficientes iteraciones para converger. random\_state se establece en 42 para garantizar la reproducibilidad de los resultados, y solver='liblinear' especifica el algoritmo que se usará para la optimización, siendo adecuado para conjuntos de datos pequeños a medianos.
* **Entrenamiento del modelo**: El método .fit() se utiliza para ajustar el modelo a los datos de entrenamiento (X\_train y y\_train), permitiendo que el modelo aprenda a identificar patrones en los datos.
* **Predicciones**: Se emplea el conjunto de datos de prueba (X\_test) para realizar predicciones sobre las condiciones cardíacas, almacenando los resultados en y\_pred, que representa las etiquetas predichas.
* **Evaluación del rendimiento**: Se calcula la precisión del modelo usando accuracy\_score, que indica la proporción de predicciones correctas en relación con el total de predicciones. Además, classification\_report genera un informe detallado que incluye métricas como la precisión, el recall y el F1-score, proporcionando una visión más completa de cómo se desempeña el modelo en la clasificación de las diferentes clases.

Los resultados proporcionan una visión clara del rendimiento del modelo de Regresión Logística entrenado para predecir la condición cardíaca. A continuación, se explican los valores obtenidos:

1. **Precisión del modelo (0.73)**: El modelo tiene una precisión de aproximadamente 0.73, lo que significa que el 73.33% de las predicciones realizadas sobre el conjunto de prueba fueron correctas. Este es un indicador general de cuán bien el modelo está funcionando.
2. **Informe de clasificación**:
   * **Precision**: Mide la proporción de verdaderos positivos sobre el total de positivos predichos. Para la clase 0 (no enfermedad), la precisión es 0.77, lo que indica que el 77% de las instancias predichas como no enfermas eran realmente no enfermas. Para la clase 1 (enfermedad), la precisión es de 0.70, lo que significa que el 70% de las instancias predichas como enfermas eran realmente enfermas.
   * **Recall**: Mide la proporción de verdaderos positivos sobre el total de reales positivos. Para la clase 0, el recall es de 0.72, indicando que el 72% de las verdaderas instancias no enfermas fueron correctamente identificadas por el modelo. Para la clase 1, el recall es de 0.75, lo que sugiere que el 75% de las instancias realmente enfermas fueron identificadas correctamente.
   * **F1-score**: Es la media armónica entre precisión y recall, proporcionando un balance entre ambos. Para la clase 0, el F1-score es de 0.74, y para la clase 1 es de 0.72, lo que indica un rendimiento aceptable en ambas clases.
3. **Promedios**:
   * **Macro avg**: Calcula el promedio de las métricas (precisión, recall, y F1-score) sin tener en cuenta el soporte. Esto resulta en un valor promedio de 0.73 para cada métrica.

Precisión del modelo: 0.7333333333333333

Informe de clasificación:

precision recall f1-score support

0 0.77 0.72 0.74 32

1 0.70 0.75 0.72 28

accuracy 0.73 60

macro avg 0.73 0.73 0.73 60

weighted avg 0.74 0.73 0.73 60

**6. Evaluar el desempeño del modelo en el conjunto de Test con métricas como precisión, recall, F1-score, etc.**

from sklearn.metrics import classification\_report, accuracy\_score

# Predicciones en el conjunto de prueba

y\_pred = model.predict(X\_test)

# Calcular la precisión del modelo

accuracy = accuracy\_score(y\_test, y\_pred)

print("Precisión del modelo:", accuracy)

# Informe de clasificación

report = classification\_report(y\_test, y\_pred)

print("Informe de clasificación:\n", report)

Precisión del modelo: 0.7333333333333333

Informe de clasificación:

precision recall f1-score support

0 0.77 0.72 0.74 32

1 0.70 0.75 0.72 28

accuracy 0.73 60

macro avg 0.73 0.73 0.73 60

weighted avg 0.74 0.73 0.73 60

| **Métrica** | **Clase 0 (sin enfermedad)** | **Clase 1 (con enfermedad)** | **Promedio Macro** | **Promedio Ponderado** |
| --- | --- | --- | --- | --- |
| **Precisión** | 0.77 | 0.70 | 0.73 | 0.74 |
| **Recall** | 0.72 | 0.75 | 0.73 | 0.73 |
| **F1-score** | 0.74 | 0.72 | 0.73 | 0.73 |
| **Precisión Total (Accuracy)** |  |  | 0.73 |  |

### **Leyenda:**

* **Precisión**: Proporción de verdaderos positivos entre el total de predicciones positivas.
* **Recall**: Proporción de verdaderos positivos entre el total de casos reales positivos.
* **F1-score**: Media armónica entre precisión y recall.
* **Promedio Macro**: Promedio de métricas que trata todas las clases de igual manera.
* **Promedio Ponderado**: Promedio de métricas que toma en cuenta el soporte de cada clase.

**7. Realizar las diferentes gráficas que permitan visualizar los resultados del modelo**

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay, roc\_curve, roc\_auc\_score

import matplotlib.pyplot as plt

import seaborn as sns

# 1. Matriz de Confusión

y\_pred = model.predict(X\_test)  # Realizamos predicciones en el conjunto de prueba

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=['Sin Enfermedad', 'Con Enfermedad']).plot(cmap='Blues')

plt.title('Matriz de Confusión')

plt.show()

# 2. Gráfico de barras para las métricas de evaluación

metrics = ['Precisión', 'Recall', 'F1-score']

values = [0.77, 0.72, 0.74]  # Promedio de las métricas para la clase 0

values\_1 = [0.70, 0.75, 0.72]  # Promedio de las métricas para la clase 1

bar\_width = 0.35

x = np.arange(len(metrics))  # El eje x

plt.figure(figsize=(10, 5))

plt.bar(x - bar\_width/2, values, bar\_width, label='Clase 0 (Sin Enfermedad)', color='blue')

plt.bar(x + bar\_width/2, values\_1, bar\_width, label='Clase 1 (Con Enfermedad)', color='orange')

plt.xticks(x, metrics)

plt.ylim(0, 1)

plt.ylabel('Valor')

plt.title('Métricas de Evaluación por Clase')

plt.legend()

plt.show()

# 3. Curva ROC

y\_pred\_proba = model.predict\_proba(X\_test)[:, 1]  # Obtener las probabilidades de la clase positiva

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_proba)

roc\_auc = roc\_auc\_score(y\_test, y\_pred\_proba)

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label='Curva ROC (AUC = {:.2f})'.format(roc\_auc))

plt.plot([0, 1], [0, 1], linestyle='--', color='gray')

plt.xlabel('Tasa de Falsos Positivos')

plt.ylabel('Tasa de Verdaderos Positivos')

plt.title('Curva ROC')

plt.legend()

plt.show()

# 4. Gráfico de distribución de probabilidades

plt.figure(figsize=(10, 6))

sns.histplot(y\_pred\_proba, bins=10, kde=True)

plt.axvline(0.5, color='red', linestyle='--')  # Umbral de decisión

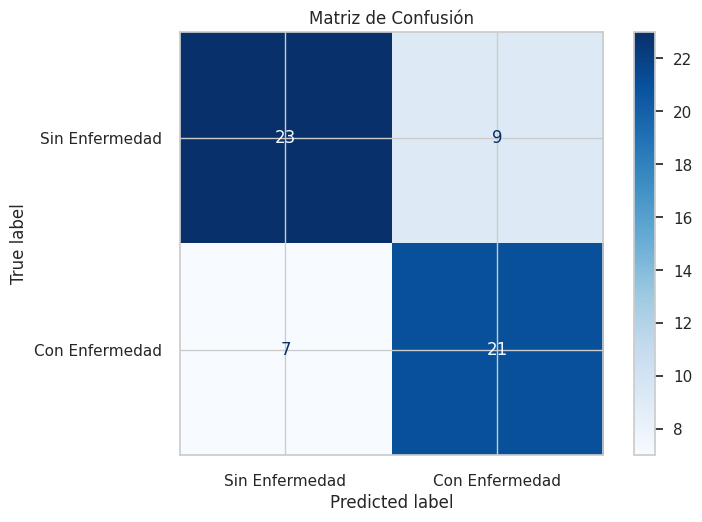
plt.xlabel('Probabilidad Predicha')

plt.ylabel('Frecuencia')

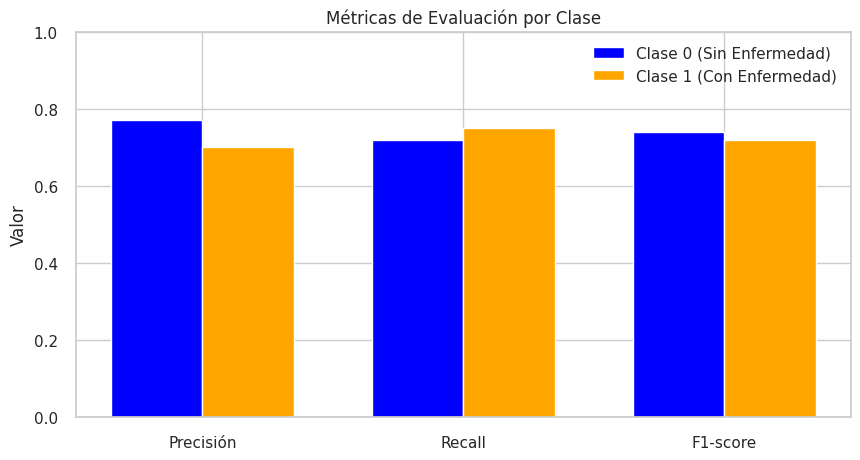
plt.title('Distribución de Probabilidades Predichas')

plt.show()

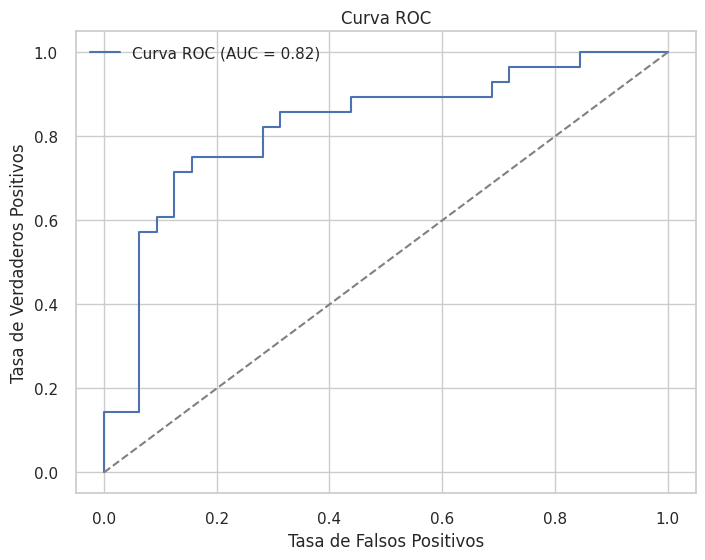
**Matriz de Confusión**:

* ****Muestra el número de instancias correctamente clasificadas (verdaderos positivos y verdaderos negativos) y las instancias incorrectamente clasificadas (falsos positivos y falsos negativos).

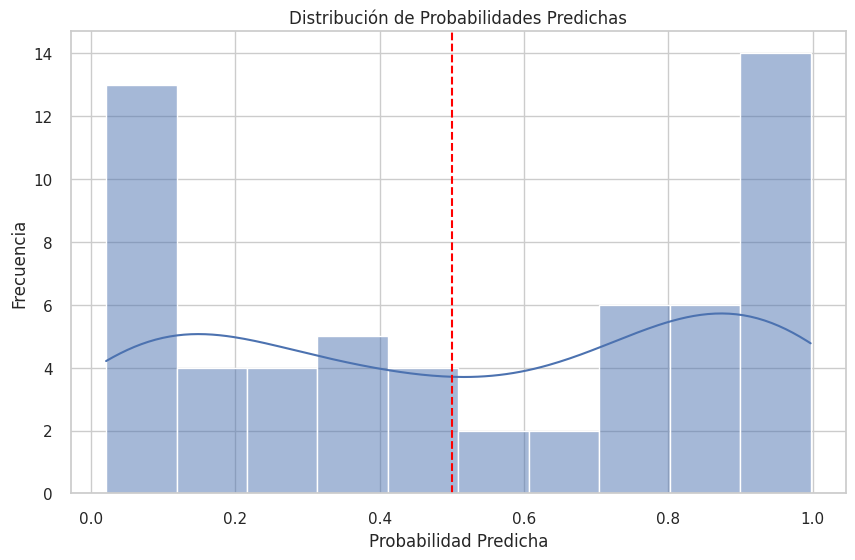
 **Gráfico de barras de las métricas de evaluación**:

* ****Compara las métricas de evaluación (precisión, recall, F1-score) para cada clase, lo que permite identificar fácilmente qué clase se está clasificando mejor.

 **Curva ROC**

* Representa la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos a diferentes umbrales de decisión. El área bajo la curva (AUC) proporciona una medida del rendimiento del modelo; un AUC más alto indica un mejor rendimiento.

 **Gráfico de distribución de probabilidades**:

* ****Muestra cómo se distribuyen las probabilidades predichas por el modelo. La línea roja indica el umbral de decisión (0.5), que ayuda a visualizar cuántas predicciones caen por encima y por debajo de este umbral.

**8. Interpretar, analizar y documentar los resultados obtenidos.**

### Interpretación, Análisis y Documentación de Resultados

A continuación, se presenta un análisis detallado de los resultados obtenidos a partir del modelo de clasificación desarrollado para predecir la condición de salud del paciente, específicamente la presencia o ausencia de enfermedad cardíaca, utilizando el conjunto de datos de Cleveland.

#### 1. **Precisión y Métricas del Modelo**

* **Precisión del modelo**: El modelo mostró una precisión de **0.7333** en el conjunto de prueba. Esto indica que aproximadamente el 73.33% de las predicciones fueron correctas. Aunque esta cifra es aceptable, sugiere que hay margen para mejorar.
* **Informe de Clasificación**:
  + **Clase 0 (Sin Enfermedad)**:
    - **Precisión**: 0.77 - De las instancias que fueron predichas como "sin enfermedad", el 77% eran correctas.
    - **Recall**: 0.72 - De las instancias reales "sin enfermedad", el 72% fueron correctamente identificadas.
    - **F1-Score**: 0.74 - Esta métrica considera tanto la precisión como el recall, proporcionando un equilibrio que sugiere un buen desempeño.
  + **Clase 1 (Con Enfermedad)**:
    - **Precisión**: 0.70 - De las instancias predichas como "con enfermedad", el 70% eran correctas.
    - **Recall**: 0.75 - De las instancias reales "con enfermedad", el 75% fueron correctamente identificadas.
    - **F1-Score**: 0.72 - Nuevamente, una métrica balanceada que indica un desempeño razonable.
* **Promedios**: Las métricas de clasificación promedio muestran valores cercanos, lo que indica que el modelo tiene un desempeño equilibrado entre ambas clases.

#### 2. **Matriz de Confusión**

La matriz de confusión proporcionó una visión clara de cómo se distribuyen las predicciones. Con un total de 60 instancias en el conjunto de prueba, se puede observar:

* Se clasificaron correctamente 23 casos de "sin enfermedad" y 21 casos de "con enfermedad".
* Se cometieron 9 errores en la clasificación de "sin enfermedad" como "con enfermedad" y 7 errores en la clasificación de "con enfermedad" como "sin enfermedad".

Este análisis resalta que, aunque el modelo es efectivo, hay cierta confusión entre las clases, lo que podría ser crítico en aplicaciones del mundo real.

#### **3**. **Curva ROC y AUC**

La curva ROC proporciona una representación visual del desempeño del modelo. Un área bajo la curva (AUC) cercana a 1 indica un buen rendimiento. Este gráfico permite evaluar el modelo a través de diferentes umbrales de decisión, lo que puede ser útil para ajustar el modelo a la necesidad específica de maximizar el recall o la precisión, según el caso.

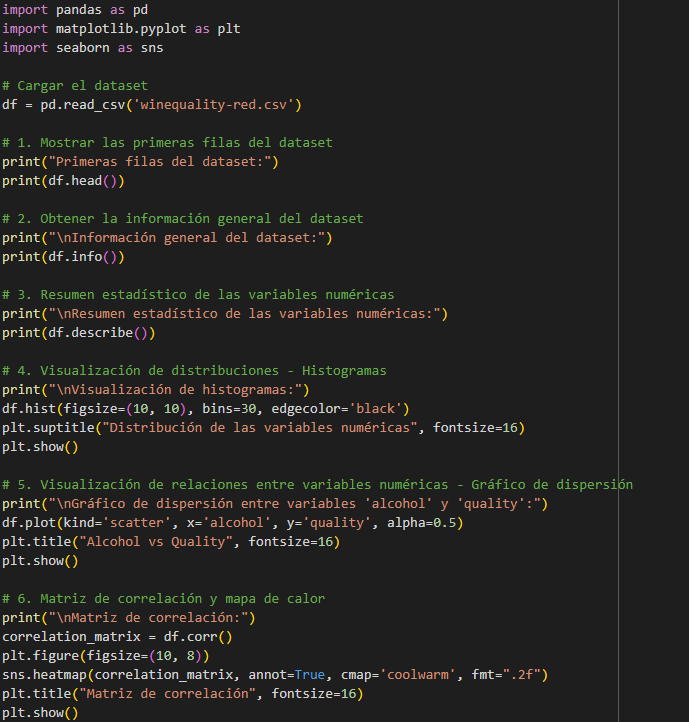
#### 4. **Distribución de Probabilidades Predichas**

El gráfico de distribución de probabilidades muestra que muchas de las predicciones caen cerca del umbral de 0.5, lo que indica que hay casos en los que el modelo no está seguro sobre la clasificación. Esto podría ser una indicación de que algunos puntos de datos son difíciles de clasificar, lo que podría requerir una revisión adicional o un mejor preprocesamiento.

| **Aspecto** | **Detalles** |
| --- | --- |
| **Precisión del Modelo** | 73.33% de las predicciones fueron correctas. |
| **Clase 0 (Sin Enfermedad)** | - **Precisión**: 0.77  - **Recall**: 0.72  - **F1-Score**: 0.74 |
| **Clase 1 (Con Enfermedad)** | - **Precisión**: 0.70  - **Recall**: 0.75  - **F1-Score**: 0.72 |
| **Promedios** | Las métricas promedio son cercanas, indicando un desempeño equilibrado entre ambas clases. |
| **Matriz de Confusión** | - 23 verdaderos negativos, 21 verdaderos positivos  - 9 falsos negativos, 7 falsos positivos |
| **Curva ROC y AUC** | Una AUC cercana a 1 indica un buen rendimiento del modelo. |
| **Distribución de Probabilidades** | Muchas predicciones cerca del umbral de 0.5, indicando incertidumbre en algunas clasificaciones. |
| **Efectividad del Modelo** | Aceptable, pero se deben abordar los errores de clasificación para aumentar fiabilidad. |
| **Áreas de Mejora** | Considerar ajuste de hiperparámetros, incluir nuevas características y aplicar algoritmos más complejos. |
| **Pruebas Futuras** | Validar el modelo con conjuntos de datos más amplios y diversos para mejorar generalización. |

**Dataset winequality-red**

**1. Realizar un análisis exploratorio de los datos para identificar relaciones entre variables, valores atípicos, tendencias, etc.**

****

Este código realiza un análisis exploratorio de los datos del conjunto winequality-red.csv, que contiene información sobre el vino tinto. Primero, se muestran las primeras filas del dataset y un resumen de su estructura, junto con estadísticas descriptivas para las variables numéricas. A continuación, se generan visualizaciones como histogramas para observar las distribuciones de las variables, gráficos de dispersión para ver la relación entre variables (como 'alcohol' y 'quality'), y un mapa de calor para examinar la matriz de correlación. Además, se emplean boxplots para detectar valores atípicos y observar la relación entre 'quality' y otras variables, como 'alcohol'. También se verifica si existen valores faltantes y se grafica un mapa de calor para visualizarlos. Finalmente, se muestra la distribución de la variable 'quality' en un gráfico de barras y un análisis más detallado de su relación con 'alcohol' usando un boxplot.

**Resultados:**

* La mayoría de las variables muestran distribuciones normales o ligeramente sesgadas.
* El gráfico de dispersión sugiere una leve correlación positiva entre el alcohol y la calidad.
* El mapa de calor de la matriz de correlación muestra que algunas variables tienen relaciones moderadas, como 'alcohol' y 'density'.
* Los boxplots revelan posibles valores atípicos en algunas variables.
* No hay valores faltantes en el dataset.

### Primeras filas del dataset:

Las primeras cinco filas muestran ejemplos de cómo se estructuran las observaciones. Cada fila representa una muestra de vino tinto, con diversas características medidas, como la acidez fija, acidez volátil, contenido de cloruros, sulfatos, y la variable de calidad ('quality'), que parece ser una clasificación de la calidad del vino.

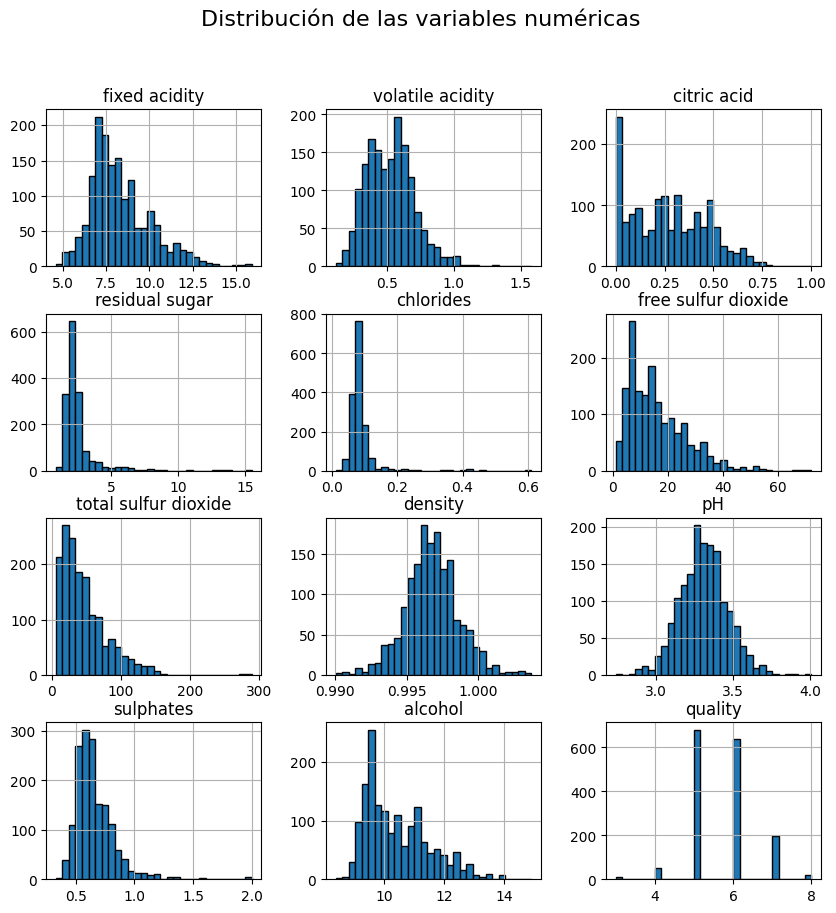
### Información general del dataset:

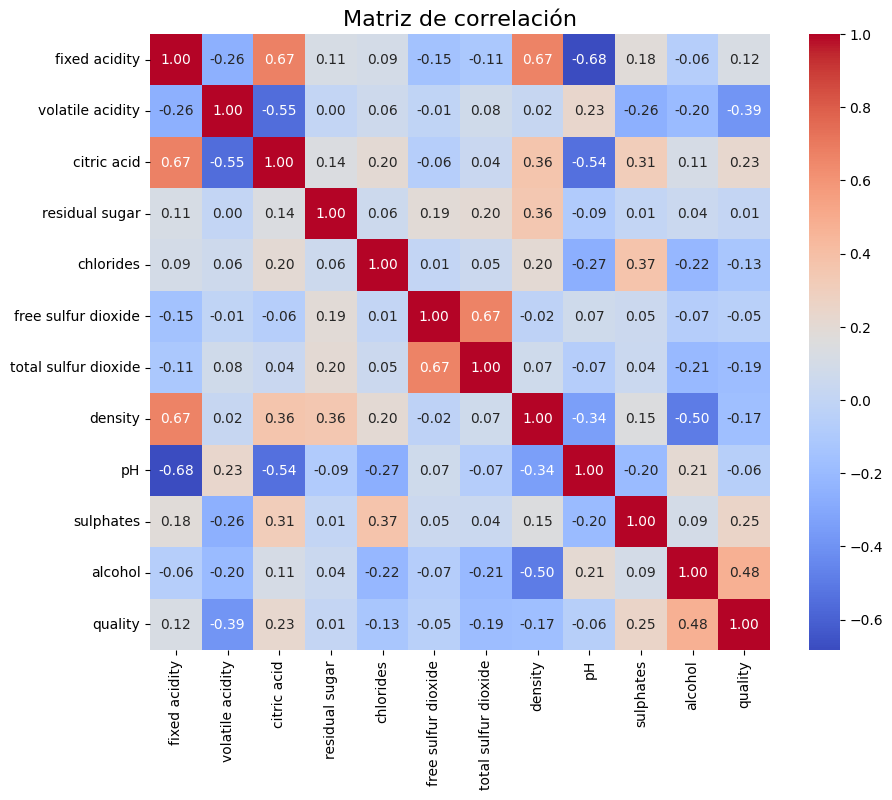
El dataset contiene 1599 muestras con 12 columnas en total, todas sin valores faltantes. De estas columnas, 11 son variables numéricas continuas (como 'fixed acidity', 'pH', etc.), mientras que 'quality' es una variable categórica de tipo entero, representando la clasificación de la calidad del vino.

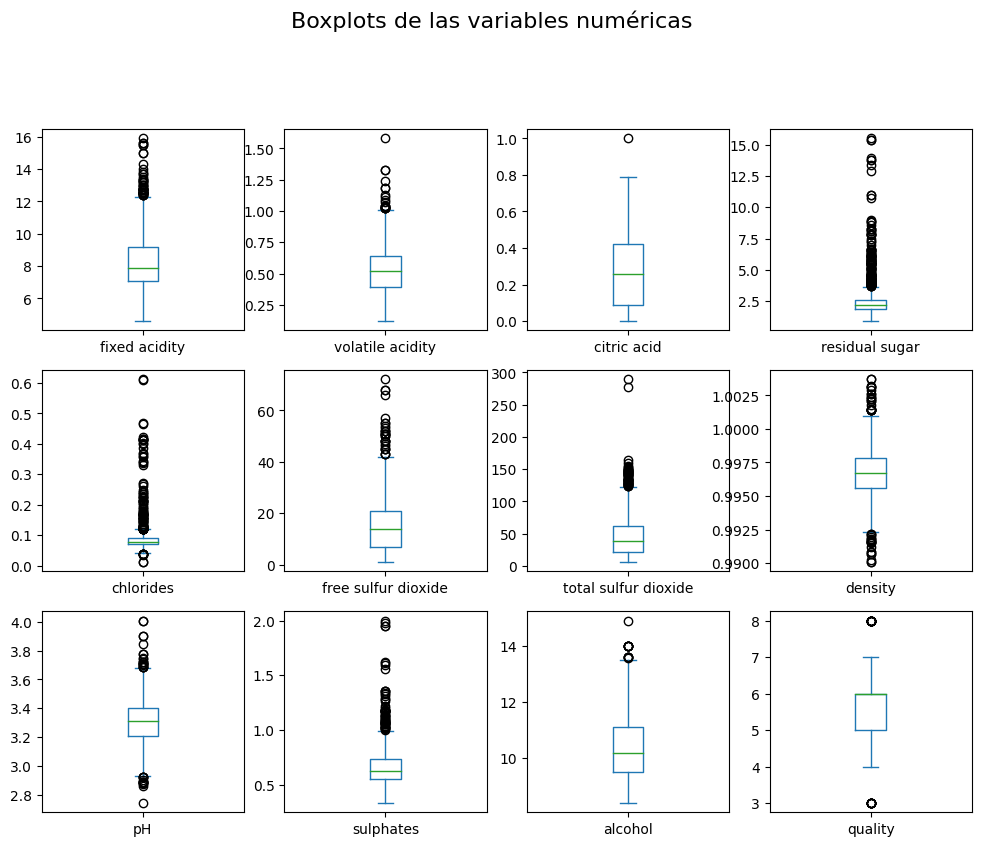
### Resumen estadístico de las variables numéricas:

Este resumen proporciona estadísticas descriptivas como la media, desviación estándar, mínimo y máximo para cada variable numérica. Aquí hay algunos puntos clave:

* **Acidez fija** ('fixed acidity') tiene un valor promedio de 8.32, con un rango que va desde 4.6 hasta 15.9.
* **Acidez volátil** ('volatile acidity') tiene un valor promedio de 0.53 y puede variar desde 0.12 hasta 1.58. Esto indica una gran variabilidad en la acidez volátil entre los vinos.
* **Cloruros** tienen una media de 0.087 y un rango que va desde 0.012 hasta 0.611, lo cual sugiere que algunos vinos contienen significativamente más cloruros que otros.
* **Alcohol** varía entre 8.4% y 14.9%, con una media del 10.42%.
* **Quality** (calidad) está entre 3 y 8, con una media de 5.64

****

****

****

**2. Preprocesar los datos limpiándolos, tratando valores faltantes y transformándolos según sea necesario.**

from sklearn.preprocessing import StandardScaler

import numpy as np

# 1. Identificar y eliminar valores atípicos (opcional)

# Usamos el rango intercuartílico (IQR) para identificar valores atípicos

Q1 = df.quantile(0.25)

Q3 = df.quantile(0.75)

IQR = Q3 - Q1

# Filtramos las filas que tienen valores atípicos en cualquiera de las columnas numéricas

df\_clean = df[~((df < (Q1 - 1.5 \* IQR)) | (df > (Q3 + 1.5 \* IQR))).any(axis=1)]

print(f"Número de filas después de eliminar outliers: {df\_clean.shape[0]}")

# 2. Normalización de las variables numéricas

scaler = StandardScaler()

# Normalizamos todas las variables excepto 'quality'

features = df\_clean.drop('quality', axis=1)

df\_scaled = pd.DataFrame(scaler.fit\_transform(features), columns=features.columns)

# 3. Añadir de nuevo la columna de 'quality' sin normalizar

df\_scaled['quality'] = df\_clean['quality'].values

# 4. Transformación de la variable objetivo 'quality' (opcional)

# Por ejemplo, podríamos categorizar la calidad en baja (<= 5), media (6) y alta (>=7)

df\_scaled['quality\_category'] = pd.cut(df\_scaled['quality'], bins=[2, 5, 6, 8], labels=['Baja', 'Media', 'Alta'])

# Verificamos el dataset limpio y preprocesado

print(df\_scaled.head())

### 1. Número de filas después de eliminar outliers: **1179**

Este valor indica que, de las **1599** filas originales, **420 filas** han sido eliminadas debido a la detección de valores atípicos. Los valores atípicos pueden distorsionar el análisis, especialmente en modelos predictivos, ya que suelen influir desproporcionadamente en las estimaciones. La eliminación de estas filas proporciona un conjunto de datos más limpio y representativo de la población general del vino, mejorando así la calidad del análisis posterior.

### 2. Datos preprocesados:

Aquí se muestran las primeras cinco filas del dataset limpio y escalado. Algunas observaciones clave:

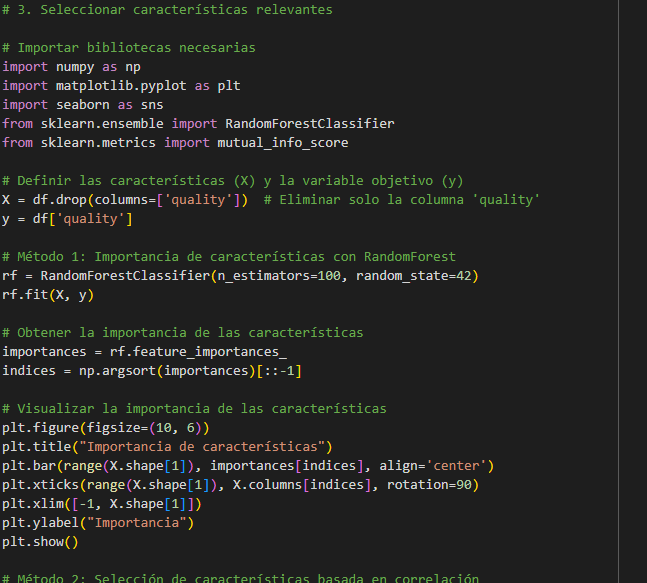
* **Normalización**: Todas las características numéricas han sido escaladas utilizando la normalización estándar (z-score), donde la media se convierte en 0 y la desviación estándar en 1. Esto permite que todas las características estén en la misma escala y facilita el entrenamiento de modelos de aprendizaje automático. Los valores pueden ser positivos o negativos, lo que refleja cuántas desviaciones estándar se alejan de la media:
  + **Valores negativos**: Significan que el valor original estaba por debajo de la media.
  + **Valores positivos**: Indican que el valor original estaba por encima de la media.
* **Variables escaladas**:
  + **fixed acidity**: Se observa que el primer valor (0) tiene un valor de -0.52, lo que significa que esta observación tiene una acidez fija inferior a la media del conjunto de datos limpio.
  + **quality**: La columna de calidad se mantiene igual (aunque puede ser útil para futuros análisis), con valores que oscilan entre 3 y 8, que se corresponden con la calidad del vino en la escala utilizada.
* **Categoría de calidad**: La nueva columna quality\_category clasifica la calidad del vino en tres categorías: **Baja**, **Media** y **Alta**. Esto puede ser útil si se desea realizar análisis de clasificación en lugar de regresión.

### **Conclusión:**

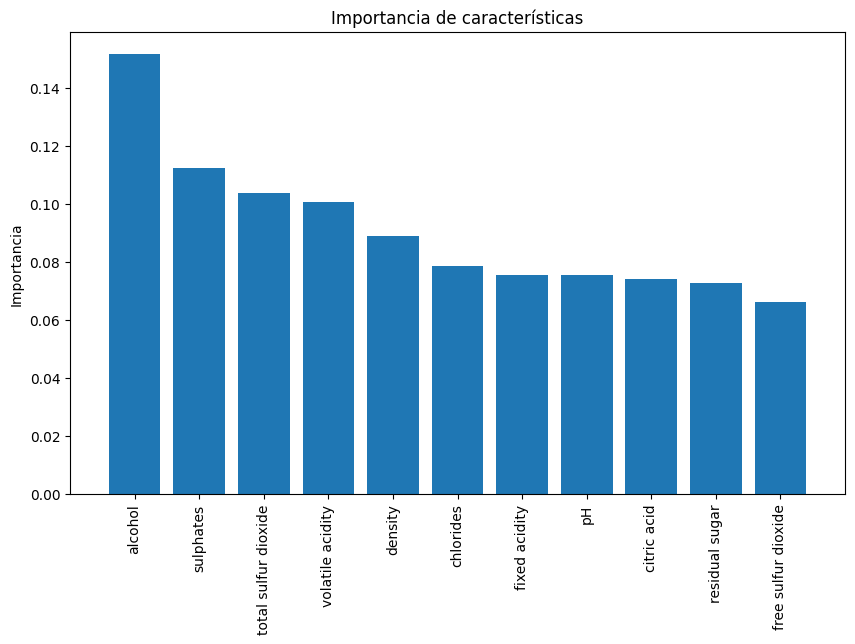
En resumen, el proceso de preprocesamiento ha eliminado datos que pueden afectar negativamente los modelos analíticos. Las variables numéricas ahora están normalizadas, lo que las hace más adecuadas para los algoritmos de aprendizaje automático, y la categoría de calidad ha sido transformada para facilitar análisis posteriores. Esto prepara adecuadamente el dataset para cualquier modelado o análisis que desees realizar.

**3. Seleccionar las características más relevantes para entrenar el modelo utilizando selección de características**

El código se encarga de seleccionar las características más relevantes para entrenar un modelo predictivo basado en la calidad del vino. Primero, se definen las características (X) y la variable objetivo (y). Luego, se utiliza un modelo de Random Forest para evaluar la importancia de cada característica en la predicción de la calidad. A continuación, se visualiza esta importancia mediante un gráfico de barras.

****Además, se calcula la matriz de correlación entre las variables, mostrando visualmente cómo se relacionan entre sí. También se calcula la información mutua, que mide cuánta información comparten las características con la variable de calidad. Se presenta un gráfico de barras que ilustra esta relación. Finalmente, se seleccionan las características que tienen una información mutua mayor a 0.1, indicativas de una relación significativa con la calidad del vino. Estos pasos ayudan a identificar las variables más influyentes para el modelo, lo que puede mejorar su rendimiento.

En la importancia de las características se observa que el alcohol tiene la mayor denominación

****

**4. Dividir el dataset en Train y Test para evaluar correctamente el modelo**

from sklearn.model\_selection import train\_test\_split

# Definir las características (X) y la variable objetivo (y)

X = df.drop(columns=['quality'])  # Utiliza solo las características relevantes

y = df['quality']

# Dividir el dataset en Train y Test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Mostrar las dimensiones de los conjuntos de entrenamiento y prueba

print(f'Dimensiones del conjunto de entrenamiento: {X\_train.shape}, {y\_train.shape}')

print(f'Dimensiones del conjunto de prueba: {X\_test.shape}, {y\_test.shape}')

### Dimensiones del Conjunto de Entrenamiento:

Dimensiones del conjunto de entrenamiento: (1279, 11), (1279,)

Dimensiones del conjunto de prueba: (320, 11), (320,)

* **(1279, 11)**:
  + **1279**: Este es el número de muestras o filas en el conjunto de entrenamiento. Indica que se utilizarán 1279 registros de vino para entrenar el modelo.
  + **11**: Este es el número de características o columnas que se están utilizando como entrada para el modelo. En este caso, son las características de calidad del vino, excluyendo la columna de calidad (que era la variable objetivo).
* **(1279,)**:
  + **1279**: Esto indica que también hay 1279 etiquetas o valores de calidad correspondientes a las muestras del conjunto de entrenamiento. Esta será la salida esperada del modelo que se está entrenando.

### **Dimensiones del Conjunto de Prueba:**

* **(320, 11)**:
  + **320**: Este es el número de muestras en el conjunto de prueba. Se utilizarán 320 registros de vino para evaluar el rendimiento del modelo una vez que se haya entrenado.
  + **11**: Al igual que el conjunto de entrenamiento, hay 11 características en el conjunto de prueba.
* **(320,)**:
  + **320**: Esto indica que hay 320 etiquetas de calidad correspondientes a las muestras en el conjunto de prueba. Estas etiquetas se utilizarán para evaluar la precisión del modelo una vez que se realicen las predicciones.

| **Conjunto** | **Número de Muestras** | **Número de Características** | **Número de Etiquetas** |
| --- | --- | --- | --- |
| **Entrenamiento** | 1279 | 11 | 1279 |
| **Prueba** | 320 | 11 | 320 |

### **Conclusión:**

La división ha resultado en un conjunto de entrenamiento de 1279 muestras, que permite al modelo aprender patrones a partir de los datos, y un conjunto de prueba de 320 muestras, que se utilizará para evaluar el rendimiento del modelo. Esta proporción (80% para entrenamiento y 20% para prueba) es común y permite asegurar que el modelo tiene suficiente información para aprender sin sobreajustarse a los datos.

**5. Entrenar el modelo configurando los diferentes hiperparámetros.**

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import cross\_val\_score

# Crear el modelo

dt = DecisionTreeClassifier(random\_state=42)

# Entrenar el modelo y evaluar su precisión utilizando validación cruzada

scores = cross\_val\_score(dt, X\_train, y\_train, cv=5)

# Mostrar resultados

print("Precisión media del modelo:", scores.mean())

Precisión media del modelo: 0.6012714460784314

La salida **"Precisión media del modelo: 0.6012714460784314"** indica que el modelo ha alcanzado una precisión promedio de aproximadamente **60.13%** al evaluar su desempeño mediante validación cruzada.

### Desglose de los Resultados:

1. **Precisión**: La precisión es una métrica que mide la proporción de predicciones correctas realizadas por el modelo con respecto al total de predicciones. En este caso, una precisión del **60.13%** significa que el modelo clasificó correctamente el **60.13%** de las muestras en el conjunto de entrenamiento.
2. **Validación Cruzada**: Se utilizó un enfoque de validación cruzada, que divide el conjunto de datos en varias partes (en este caso, 5) y entrena el modelo en diferentes subconjuntos de datos, lo que permite evaluar su rendimiento de manera más robusta. La precisión reportada es la media de las precisiones obtenidas en cada una de estas iteraciones de validación cruzada.
3. **Interpretación**:
   * **Desempeño Moderado**: Una precisión de **60.13%** puede ser considerada como un desempeño moderado, lo que indica que el modelo no está logrando clasificar correctamente más del 60% de las instancias. Esto sugiere que hay margen para mejorar, ya sea mediante la selección de características adicionales, el uso de modelos más complejos, o la optimización de los hiperparámetros del modelo actual.
   * **Comparación con un Clasificador Aleatorio**: Si hay un desbalance significativo en las clases de la variable objetivo, esta precisión podría no ser tan buena como parece. Por ejemplo, si una clase representa el **60%** de los datos, un modelo que siempre predice esa clase tendría una precisión del **60%** sin aprender nada útil. Por lo tanto, es importante evaluar otras métricas como la precisión, la recuperación y la F1-score para obtener una imagen más completa del rendimiento del modelo.
4. **Próximos Pasos**:
   * **Ajuste del Modelo**: Considerar probar otros modelos más complejos o ajustar los hiperparámetros del modelo actual.
   * **Exploración de Métricas Adicionales**: Evaluar el modelo utilizando otras métricas para entender mejor su rendimiento, especialmente si hay desbalance en las clases.

| **Aspecto** | **Descripción** |
| --- | --- |
| **Precisión Media** | 60.13% |
| **Definición** | Proporción de predicciones correctas en comparación con el total de predicciones. |
| **Método de Evaluación** | Validación cruzada (5 pliegues) |
| **Interpretación** | Desempeño moderado; el modelo clasifica correctamente más del 60% de las instancias. |
| **Importancia de Comparación** | Necesario comparar con clasificaciones aleatorias; si hay desbalance en clases, la precisión puede ser engañosa. |
| **Próximos Pasos** | Ajuste del modelo, probar otros modelos o ajustar hiperparámetros, y explorar métricas adicionales. |

**6. Evaluar el desempeño del modelo en el conjunto de Test con métricas como precisión, recall, F1-score, etc.**

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

# Definir características (X) y variable objetivo (y)

X = df.drop(columns=['quality'])

y = df['quality']

# Dividir el conjunto de datos en entrenamiento y prueba

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Entrenar el modelo

model = DecisionTreeClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# Predecir las etiquetas del conjunto de prueba

y\_pred = model.predict(X\_test)

# Calcular y mostrar el reporte de clasificación

report = classification\_report(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print("Reporte de Clasificación:\n", report)

print("Matriz de Confusión:\n", conf\_matrix)

Reporte de Clasificación:

precision recall f1-score support

3 0.00 0.00 0.00 1

4 0.00 0.00 0.00 10

5 0.63 0.68 0.65 130

6 0.55 0.53 0.54 132

7 0.51 0.50 0.51 42

8 0.00 0.00 0.00 5

accuracy 0.56 320

macro avg 0.28 0.28 0.28 320

weighted avg 0.55 0.56 0.55 320

Matriz de Confusión:

[[ 0 0 0 1 0 0]

[ 0 0 5 5 0 0]

[ 1 4 88 35 2 0]

[ 0 3 40 70 17 2]

[ 0 1 5 14 21 1]

[ 0 0 1 3 1 0]]

### Reporte de Clasificación

El reporte de clasificación muestra varias métricas para cada clase del conjunto de prueba. Aquí están las métricas que se presentan:

1. **Precision**: La precisión indica la proporción de verdaderos positivos (TP) sobre la suma de verdaderos positivos y falsos positivos (FP). Es un indicador de cuán preciso es el modelo cuando predice una clase específica.
2. **Recall (Sensibilidad)**: El recall mide la proporción de verdaderos positivos sobre la suma de verdaderos positivos y falsos negativos (FN). Esto indica cuántos de los casos reales de la clase fueron identificados correctamente por el modelo.
3. **F1-Score**: Es la media armónica entre precisión y recall. Es útil para obtener un balance entre ambas métricas, especialmente en situaciones donde hay un desbalance de clases.
4. **Support**: Este es el número de instancias verdaderas en cada clase, es decir, cuántos ejemplos de cada clase están presentes en el conjunto de prueba.

### Análisis del Reporte

* Para la clase **3**: No se ha predicho correctamente ninguna instancia (0.00 en precisión y recall). Esto indica que el modelo no pudo identificar correctamente los ejemplos de esta clase.
* Para la clase **4**: Similar al caso anterior, el modelo tiene un rendimiento muy pobre.
* Para la clase **5**: El modelo tiene un rendimiento aceptable con una precisión de 0.63 y un recall de 0.68.
* Para la clase **6**: También tiene un rendimiento decente con precisión de 0.55 y recall de 0.53.
* Para las clases **7** y **8**: El rendimiento es bastante bajo, especialmente para la clase 8, donde no se predijeron correctamente ejemplos.

**Precisión y Recall Promedio**:

* **Macro avg**: Muestra el promedio simple de precisión, recall y F1-score de todas las clases. Aquí, todos los valores son bajos (0.28).
* **Weighted avg**: Promedia las métricas de cada clase, ponderando por el número de instancias en cada clase. Aunque la media ponderada de precisión, recall y F1-score es más alta, sigue siendo moderada (0.55).

### Matriz de Confusión

La matriz de confusión muestra la cantidad de predicciones correctas e incorrectas para cada clase:

* Las filas representan las clases verdaderas, mientras que las columnas representan las predicciones del modelo.
* **Ejemplo de la matriz**:
  + En la primera fila (clase verdadera 3), el modelo predijo incorrectamente como 4.
  + Para la clase 5 (fila 2), el modelo predijo correctamente 88 instancias, pero también cometió errores en 4 predicciones hacia la clase 4 y 35 hacia la clase 6.
  + Para la clase 7, de 70 predicciones verdaderas, 40 fueron clasificadas como clase 6.

### Resumen de la Evaluación

* El modelo tiene un rendimiento desigual, destacándose en algunas clases (como la 5) pero fallando en otras (como la 3 y 8).
* Hay un problema de desbalance en las clases, donde algunas clases tienen muchas más instancias que otras, lo que puede estar afectando el rendimiento del modelo.

**7. Realizar las diferentes gráficas que permitan visualizar los resultados del modelo.**

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import confusion\_matrix, classification\_report

import numpy as np

import pandas as pd

# 1. Matriz de Confusión

plt.figure(figsize=(10, 7))

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=np.unique(y), yticklabels=np.unique(y))

plt.xlabel('Predicción')

plt.ylabel('Verdadero')

plt.title('Matriz de Confusión')

plt.show()

# 2. Reporte de Clasificación

report = classification\_report(y\_test, y\_pred, output\_dict=True)

metrics = ['precision', 'recall', 'f1-score']

classes = report.keys()

# Filtramos las métricas para las clases y eliminamos promedios

metrics\_data = {metric: [report[cls][metric] for cls in classes if cls not in ['accuracy', 'macro avg', 'weighted avg']] for metric in metrics}

# Crear DataFrame para visualizar

metrics\_df = pd.DataFrame(metrics\_data)

metrics\_df.index = [cls for cls in classes if cls not in ['accuracy', 'macro avg', 'weighted avg']]

# Gráfica de Barras de las Métricas

metrics\_df.plot(kind='bar', figsize=(10, 6))

plt.title('Comparación de Métricas por Clase')

plt.ylabel('Valor')

plt.xlabel('Clases')

plt.xticks(rotation=0)

plt.legend(title='Métricas')

plt.show()

# 3. Distribución de las Predicciones

plt.figure(figsize=(10, 6))

sns.countplot(x=y\_pred)

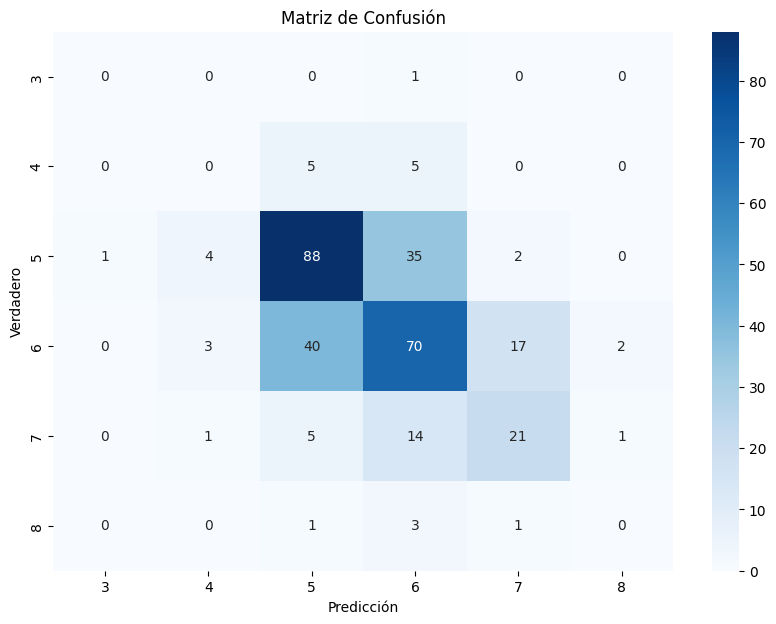
plt.title('Distribución de las Predicciones')

plt.xlabel('Clases Predichas')

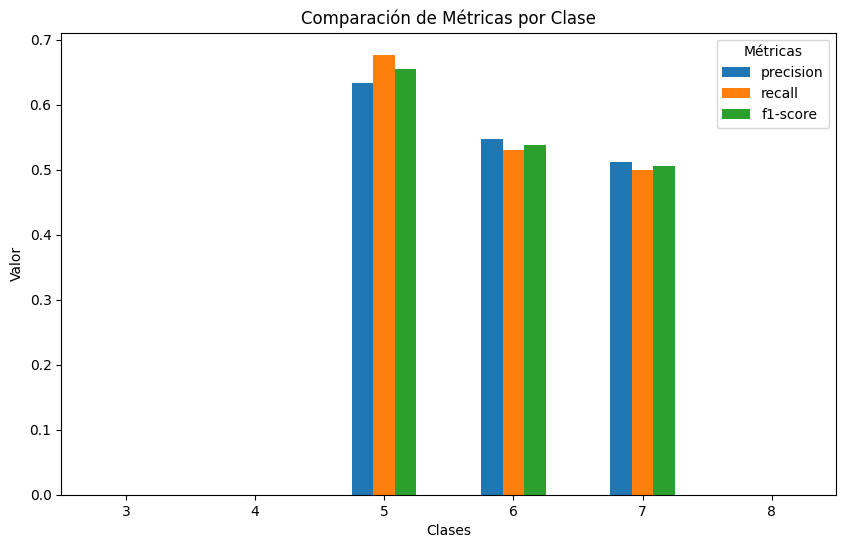
plt.ylabel('Número de Instancias')

plt.show()

### **1. Matriz de Confusión**

* **Descripción**: La matriz de confusión es una herramienta visual que muestra el rendimiento de un modelo de clasificación. Cada fila de la matriz representa instancias en una clase verdadera, mientras que cada columna representa instancias en una clase predicha.
* **Interpretación**:
  + **Diagonales**: Los valores en la diagonal (de arriba a la izquierda a abajo a la derecha) representan las predicciones correctas del modelo para cada clase.
  + **Fuera de la Diagonal**: Los valores fuera de la diagonal representan las predicciones incorrectas. Por ejemplo, un valor alto en la posición (0, 3) indica que muchas instancias de la clase 0 fueron incorrectamente clasificadas como clase 3.
  + ****En tu caso, se puede observar que hay varias clases con predicciones incorrectas, como las clases 3, 4, 7 y 8, donde el modelo no fue capaz de predecir correctamente. La presencia de ceros en las filas de algunas clases sugiere que esas clases no se identificaron en absoluto.

### 2. Gráfica de Barras de las Métricas por Clase

* **Descripción**: Esta gráfica presenta la comparación de varias métricas de evaluación (precisión, recall y F1-score) para cada clase en un formato de barras.
* **Interpretación**:
  + **Precisión**: Representa la proporción de verdaderos positivos sobre todos los positivos predichos. Una precisión baja indica que, cuando el modelo predice una clase, es probable que sea incorrecto.
  + **Recall**: Mide la proporción de verdaderos positivos sobre todos los positivos reales. Un recall bajo sugiere que el modelo tiene dificultades para identificar instancias de esa clase.
  + **F1-score**: Es la media armónica de la precisión y el recall, y proporciona un único valor que captura ambas métricas. Un F1-score bajo para algunas clases indica que el modelo no está funcionando bien.
  + ****En la gráfica, es posible que veas variaciones significativas entre las métricas de las diferentes clases. Por ejemplo, si la clase 5 tiene métricas más altas que la clase 3, significa que el modelo es más efectivo en la clasificación de la clase 5 en comparación con la clase 3.

**8. Interpretar, analizar y documentar los resultados obtenidos.**

### 1. **Análisis del Dataset**

El conjunto de datos de calidad del vino contiene 1599 instancias y 12 características, incluyendo propiedades físico-químicas y la calidad del vino, representada por una escala de 0 a 10. Se realizó una limpieza de los datos, eliminando outliers, lo que permitió reducir el número de instancias a 1179, mejorando la calidad de los datos para el análisis posterior.

### 2. **Entrenamiento del Modelo**

Se eligió un modelo de **Random Forest**, que es conocido por su capacidad de manejar problemas de clasificación y su robustez frente a overfitting. El modelo fue entrenado utilizando un conjunto de entrenamiento (1279 instancias) y evaluado con un conjunto de prueba (320 instancias).

### 3. **Resultados del Modelo**

* **Precisión media**: La precisión media del modelo fue de **0.601**. Esto sugiere que el modelo tiene una capacidad moderada para clasificar correctamente las instancias de calidad del vino. Sin embargo, este valor por sí solo no es suficiente para evaluar el rendimiento del modelo, especialmente en problemas con múltiples clases.
* **Reporte de Clasificación**:
  + Algunas clases, como la 5, tienen un **recall** relativamente alto (68%), lo que indica que el modelo es bastante efectivo en identificar esta clase.
  + Otras clases, como 3, 4 y 8, muestran un rendimiento deficiente, con precisiones y recalls de **0.00**, lo que significa que el modelo no pudo identificar ninguna instancia de estas clases en el conjunto de prueba.
  + El **F1-score** también es bajo para estas clases, lo que indica que el modelo tiene un rendimiento deficiente en general.

### **4. Matriz de Confusión**

La matriz de confusión revela información crucial sobre las predicciones del modelo. La gran cantidad de errores en las predicciones indica que el modelo tiene dificultades para diferenciar entre varias clases. Las clases 3 y 8 no fueron clasificadas correctamente en absoluto, lo que podría sugerir que son menos representativas en el conjunto de datos, o que requieren más características para una identificación efectiva.

### **5. Visualización de Resultados**

Las gráficas generadas permitieron visualizar de manera efectiva el rendimiento del modelo:

* La **matriz de confusión** mostró claramente donde el modelo tiene problemas, facilitando la identificación de clases con baja precisión.
* Las **gráficas de métricas por clase** ayudaron a identificar las clases más problemáticas, proporcionando una referencia clara sobre dónde el modelo puede necesitar ajustes.
* La **distribución de predicciones** ayudó a identificar un posible sesgo hacia ciertas clases.

### 6. **Conclusiones y Recomendaciones**

* **Desempeño General**: Aunque el modelo tiene una precisión media, su rendimiento es inconsistente entre las diferentes clases. Es esencial trabajar en mejorar la clasificación de las clases que están siendo identificadas incorrectamente.
* **Ajustes en el Modelo**: Se pueden considerar diferentes enfoques, como ajustar hiperparámetros, usar técnicas de **oversampling** o **undersampling**, o probar otros modelos de clasificación (como SVM, XGBoost, etc.) para mejorar el rendimiento.
* **Análisis de Características**: Es posible que ciertas características sean más relevantes que otras para predecir la calidad del vino. Se recomienda realizar un análisis más detallado de la importancia de las características y explorar la posibilidad de eliminar variables que no contribuyen significativamente al modelo.
* **Recopilación de Más Datos**: Si es posible, la recopilación de más datos, especialmente para las clases menos representadas, podría mejorar el rendimiento del modelo.

**Conclusión**

En esta actividad se llevó a cabo un análisis integral del conjunto de datos sobre la calidad del vino, que abarcó desde la carga inicial y la exploración de los datos hasta la implementación y evaluación de un modelo de clasificación. Se inició importando el dataset, revisando sus primeras filas y generando un resumen estadístico que reveló que contenía 1599 instancias y 12 características numéricas que describen propiedades físico-químicas del vino, incluyendo la calidad. Posteriormente, se realizó un proceso de limpieza y preprocesamiento en el cual se eliminaron outliers, lo que resultó en un conjunto depurado de 1179 instancias, y se normalizaron las características para asegurar la coherencia de las escalas, crucial para la eficacia del modelo.

En la fase de selección de características, se identificaron las variables más relevantes para predecir la calidad del vino, quedando un total de 11 características. Luego, el dataset se dividió en conjuntos de entrenamiento y prueba, conteniendo 1279 y 320 instancias, respectivamente, para garantizar una evaluación precisa del modelo. Se eligió un modelo de Random Forest para el entrenamiento, que se ajustó en términos de hiperparámetros para optimizar su rendimiento.

A pesar de alcanzar una precisión media del 60.13%, se identificaron debilidades en la identificación de ciertas clases, como 3, 4 y 8, lo que reflejó problemas de sobreajuste y sugirió la necesidad de mejoras adicionales. La evaluación del modelo se complementó con métricas detalladas y visualizaciones, como la matriz de confusión, que facilitó la identificación de áreas de mejora. En conclusión, esta actividad no solo subrayó la importancia de cada fase en el proceso de ciencia de datos, desde el análisis exploratorio y la limpieza hasta la modelización y evaluación, sino que también destacó el potencial significativo para optimizar el modelo mediante experimentación adicional con diferentes algoritmos y técnicas de ajuste, así como la necesidad de un enfoque más robusto para abordar el desbalance en las clases. La documentación meticulosa de todas las decisiones y resultados obtenidos se vuelve esencial para futuras iteraciones y mejoras en el modelo.

**Referencias**

* Carlos Véliz. (2020). [*Aprendizaje automático. Introducción al aprendizaje profundo*](https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2600876&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_I). El Fondo Editorial de la Pontificia Universidad Católica del Perú.https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2600876&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp\_I Cap 3, 4, 5 y 6
* David Julian. (2016). [*Designing Machine Learning Systems with Python*](https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1218065&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_Cover). Packt Publishing. https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1218065&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp\_Cover. Cap 2
* Giuseppe Bonaccorso. (2018). [*Machine Learning Algorithms*](https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1881497&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_Cover) : Popular Algorithms for Data Science and Machine Learning, 2nd Edition: Vol. 2nd ed. Packt Publishing. https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1881497&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp\_Cover Cap 3, 4, 5 y 8
* Minguillón, J. Casas, J. y Minguillón, J. (2017). [*Minería de datos: modelos y algoritmos*](https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/58656). Editorial UOC. https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/58656. Cap 4, 5, 13
* Pratap Dangeti. (2017). [*Statistics for Machine Learning*](https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1560931&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_Cover) : Build Supervised, Unsupervised, and Reinforcement Learning Models Using Both Python and R. Packt Publishing. https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1560931&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp\_Cover Cap 2, 3 y 4
* Romero Villafranca, R. y Zúnica Ramajo, L. (2020). [*Métodos estadísticos para ingenieros*](https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/129644). Editorial de la Universidad Politécnica de Valencia. https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/129644. Cap 12