

TAREA 3 – ALGORITMOS DE APRENDIZAJE SUPERVISADO

David Esteban Lasso Ordóñez

Universidad Nacional Abierta y a Distancia

Análisis de Datos (202016908_42)

Breyner Alexander Parra

2024

TABLA DE CONTENIDO

1	INTRODUCCIÓN	3
2	OBJETIVO.....	4
3	CUADRO SINÓPTICO.....	5
4	LISTADO DE DEFINICIONES	6
5	INSTALACIÓN DE ANACONDA	9
6	DISEÑO DE MODELOS PREDICTIVOS	11
6.1	Realizar un análisis exploratorio	11
6.2	Preprocesamiento de datos.	13
6.2.1	Limpieza de datos.....	13
6.2.2	Tratamiento de datos	14
6.2.3	Transformación de datos	15
6.3	Selección de características relevantes	16
6.3.1	Vehicle Dataset	16
6.3.2	Heart Diseases Dataset.....	16
6.3.3	Wine Quality Dataset	17
6.4	Division del dataset (Wine Quality)	17
6.5	Entrenamiento de modelo	19
6.6	Evaluación del modelo.....	19
6.7	Gráfico del árbol de decisión.....	20
6.8	Interpretación de resultados	20
7	BIBLIOGRAFÍA.....	22
8	ANOTACIÓN DE ENTREGA	23

1 INTRODUCCIÓN

En este trabajo se aplicaron los algoritmos de machine learning supervisado con el fin de entrenar un modelo mediante el procesamiento de datos utilizando aprendizaje autónomo para limpiar el conjuntos de datos, así como para detectar y corregir valores faltantes y atípicos que permitan diseñar el modelo predictivo.

1.1 Código de GitHub

https://github.com/ingDavidLasso/tarea_3_david_lasso_G-42.git

2 OBJETIVO

Aplicar algoritmos de Machine Learning supervisado según el problema, empleando métodos de modelado predictivo como regresión y clasificación.

3 CUADRO SINÓPTICO

Elaboración de cuadro sinóptico

A continuación, se presenta el cuadro como sinóptico propuesto con el fin de otorgar una guía visual estructurada para entender y organizar los diferentes modelos de aprendizaje supervisado.

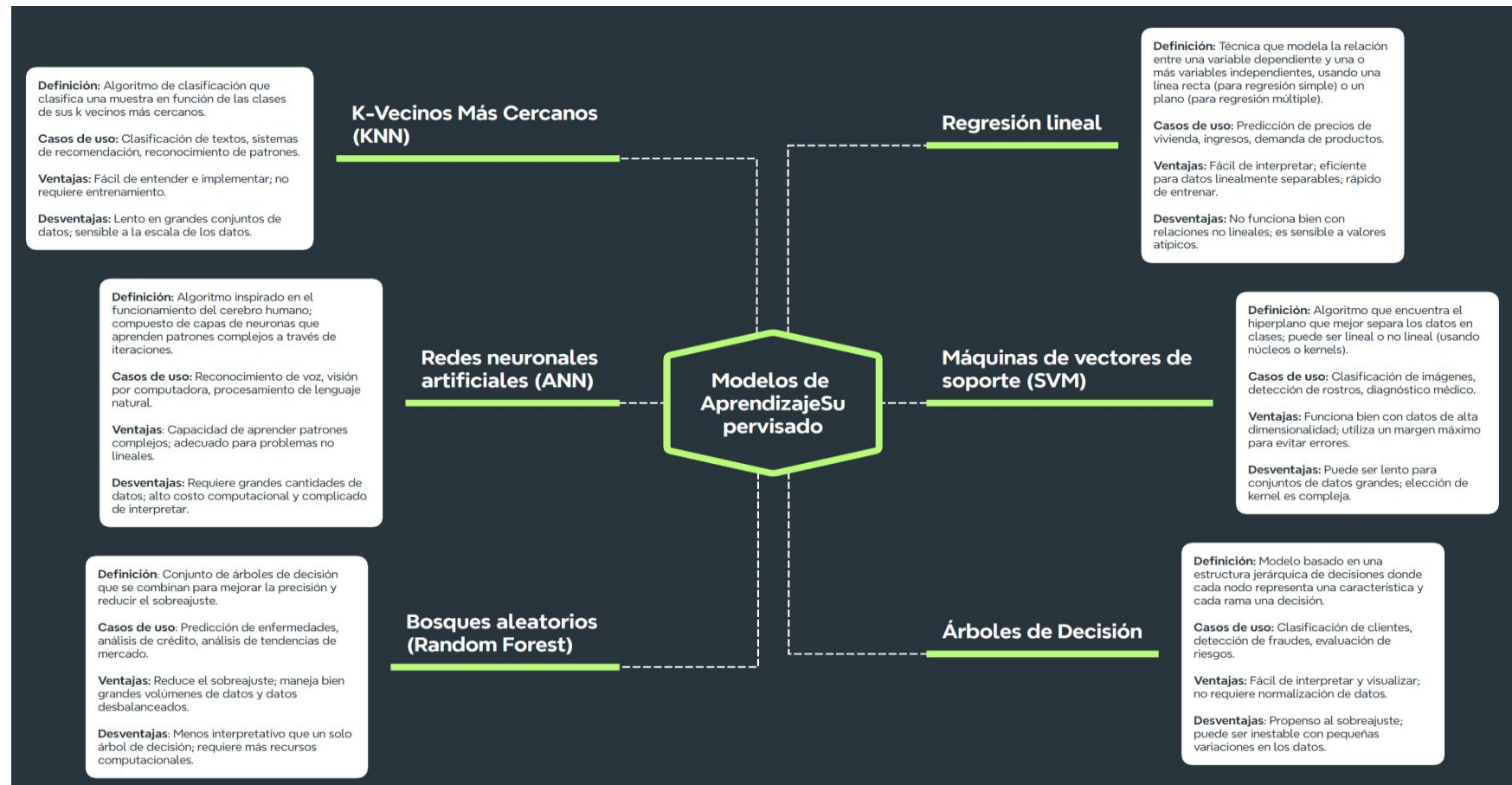


Figura 1. Cuadro sinóptico de modelos de aprendizaje supervisado.

4 LISTADO DE DEFINICIONES

Conforme a la necesidad de conocimiento de las variables mencionadas en la presente actividad, se propone el siguiente listado de definiciones:

1. **Datos de Train (Entrenamiento):** Subconjunto de datos usado para ajustar el modelo. Estos datos permiten que el modelo aprenda los patrones y relaciones en el conjunto, pero no son usados para evaluar el rendimiento del modelo en nuevos datos (de los Santos, 2022).
2. **Datos de Validation (Validación):** Conjunto de datos usado para ajustar hiperparámetros y prevenir el sobreajuste. No se utiliza para el entrenamiento directo del modelo, sino para evaluar su rendimiento mientras se optimizan los parámetros (Brownlee, 2020).
3. **Datos de Test (Prueba):** Conjunto de datos independientes utilizados exclusivamente para evaluar el rendimiento final del modelo. Estos datos no se ven durante el entrenamiento ni la validación, y ayudan a medir la capacidad de generalización del modelo.
4. **GridSearchCV:** Técnica de optimización de hiperparámetros que permite encontrar la mejor combinación de parámetros para un modelo mediante una búsqueda exhaustiva en un espacio definido. Usa validación cruzada para evaluar cada combinación de parámetros (scikit, s. f.).
5. **One Hot Encoding:** Método de codificación de variables categóricas en el que cada categoría única se convierte en una columna binaria (0 o 1). Es útil cuando se desea representar variables categóricas de manera que el modelo pueda procesarlas (GeeksforGeeks, 2019).
6. **Matriz de Confusión:** Tabla que permite evaluar el rendimiento de un modelo de clasificación, mostrando la cantidad de predicciones verdaderas y falsas para cada clase. Incluye valores como Verdaderos Positivos (VP), Verdaderos Negativos (VN), Falsos Positivos (FP), y Falsos Negativos (FN) (Arce, 2019).

7. **Precisión (Precisión):** Métrica que calcula la proporción de Verdaderos Positivos entre todos los casos positivos predichos. Mide la exactitud de las predicciones positivas.

$$Precision = \frac{VP}{VP + FP}$$

8. **Accuracy (Exactitud):** Proporción de predicciones correctas entre todas las predicciones realizadas. Evalúa qué tan frecuentemente el modelo clasifica correctamente.

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN}$$

9. **Specificity (Especificidad):** Métrica que mide la capacidad del modelo para identificar correctamente los verdaderos negativos. Calcula la proporción de Verdaderos Negativos entre los negativos reales.

$$Especificidad = \frac{VN}{VN + FP}$$

10. **Recall (Sensibilidad o Tasa de Verdaderos Positivos):** Métrica que calcula la proporción de Verdaderos Positivos entre todos los casos positivos reales. Evalúa la capacidad del modelo para detectar correctamente los casos positivos.

$$Recall = \frac{VP}{VP + FN}$$

11. **F1 Score:** Métrica que representa el balance entre Precisión y Recall. Es útil cuando se busca una armonía entre estas dos métricas, especialmente en clases desbalanceadas.

$$F1\ Score = 2 \times \frac{precision \times recall}{precision + recall}$$

12. **Curva ROC:** Gráfica que representa el rendimiento de un modelo de clasificación al calcular diferentes umbrales. Muestra la Tasa de Verdaderos Positivos (Recall) frente a la Tasa de Falsos Positivos, y el Área bajo la Curva (AUC) se utiliza para evaluar el rendimiento general del modelo.

13. **R cuadrado (Coeficiente de Determinación):** Métrica que indica la proporción de la variabilidad en la variable dependiente que puede ser explicada por el modelo. Es útil para modelos de regresión: $R^2 = 1 - \frac{\text{suma de los errores al cuadrado}}{\text{suma total de cuadrados}}$.

Un valor R^2 cercano a 1 indica un buen ajuste del modelo a los datos.

5 INSTALACIÓN DE ANACONDA

Inicialmente accedemos al sitio oficial de Anaconda, en este procedemos a realizar la descarga respecto al sistema operativo que estamos manejando. En este caso Windows.

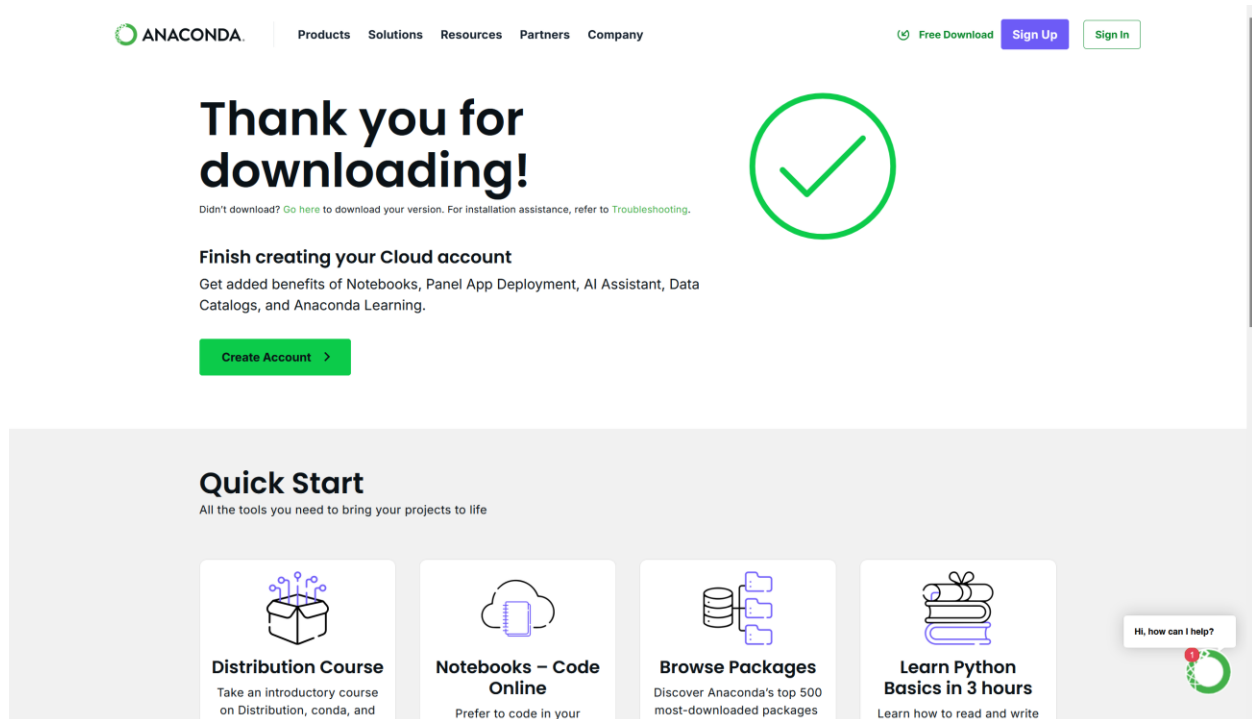


Figura 2. Inicio de descarga de Anaconda.

Una vez el instalador de Anaconda ha se ha descargado, se podrá continuar con la instalación de este.

Debido a que nuestro interés esta en ingresar en el navegador de Anaconda, lanzamos como aplicación inicial esta vista y desde aquí procederemos a enviar desde nuestro servidor generado, el Jupyter Notebook. Donde podremos continuar con los comandos para la lectura y demás procesos que se realizaran en el proceso de aprendizaje supervisado con los data sets de Kaggle.

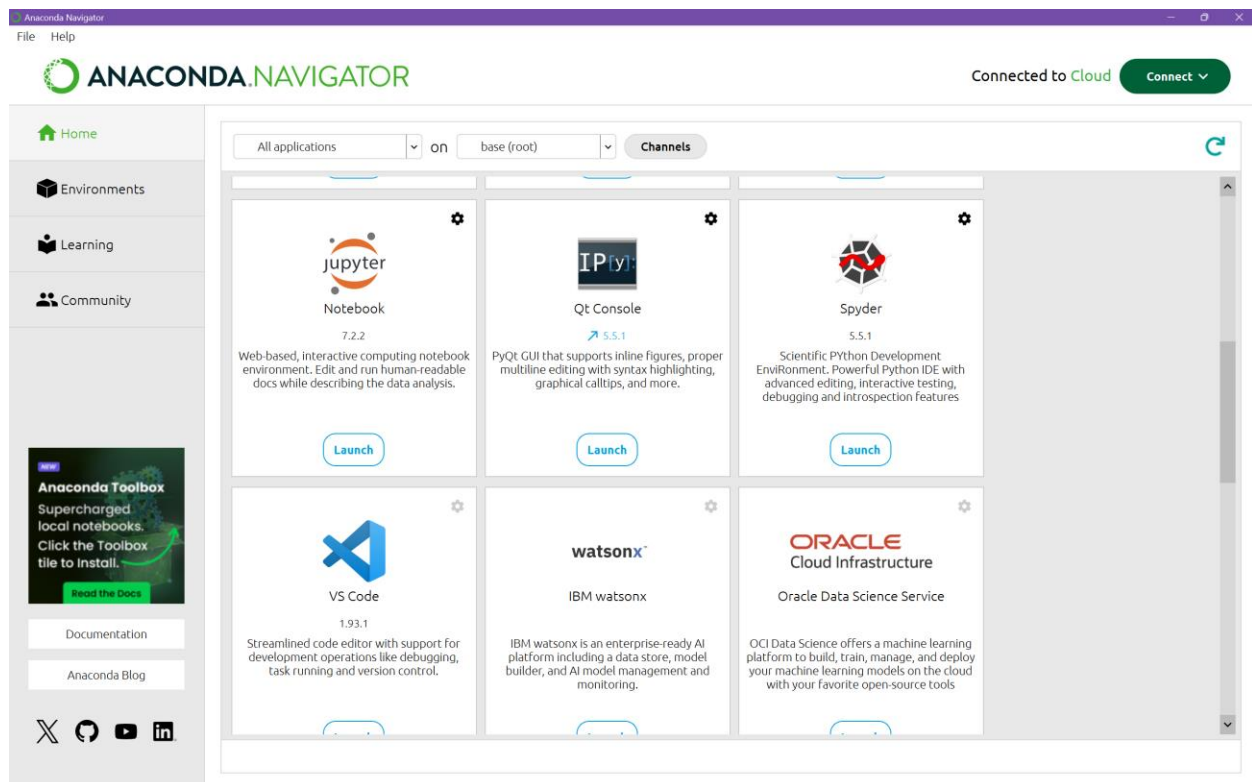


Figura 3. Navegador de anaconda en funcionamiento.

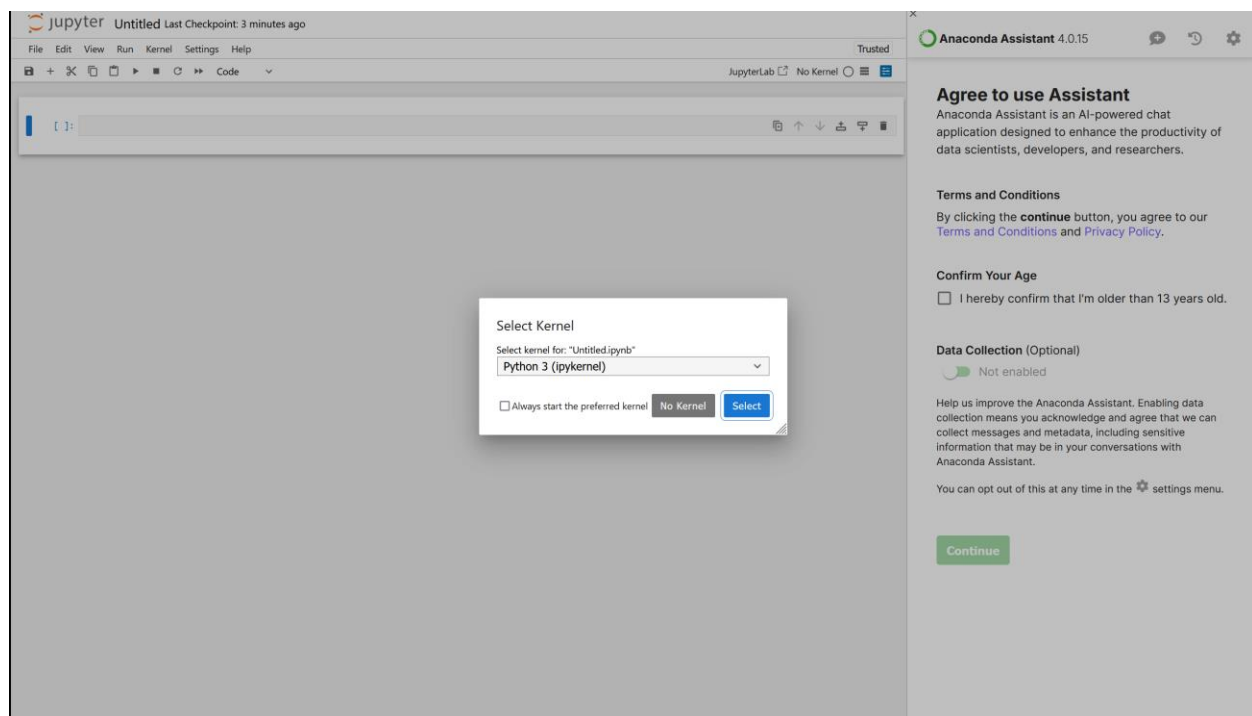


Figura 4. Lanzamiento de Jupyter Notebook correcto.

6 DISEÑO DE MODELOS PREDICTIVOS

6.1 Realizar un análisis exploratorio

El análisis exploratorio de los datos nos ayuda a conocer el contenido del dataset y ver qué patrones o problemas pueden influir en el modelo. Para este proceso cargamos un comando inicial que carga las librerías que necesitamos y también los dataset. Una vez cargados imprime la primera fila de cada uno para asegurarnos del correcto funcionamiento de estos, así:

The screenshot shows a Jupyter Notebook interface with a code cell containing Python code to load and display the first rows of three datasets: Vehicle, Heart Disease, and Wine Quality. The code is as follows:

```
[7]: # Importamos librerías
import pandas as pd

# Cargamos los datasets
vehicle_data = pd.read_csv("D:/Universities/UNAD/semestres/Octavo semestre/data analysis/tarea 3/Anexo 2 - Dataset Vehicle/car data.csv")
heart_disease_data = pd.read_csv("D:/Universities/UNAD/semestres/Octavo semestre/data analysis/tarea 3/Anexo 3 - Dataset Heart Disease Cleveland UCI/heart.csv")
wine_quality_data = pd.read_csv("D:/Universities/UNAD/semestres/Octavo semestre/data analysis/tarea 3/Anexo 4 - Dataset Red Wine Quality/winequality-red.csv")

# Imprimos la primera fila
print("Vehicle Dataset:")
display(vehicle_data.head())

print("Heart Disease Dataset:")
display(heart_disease_data.head())

print("Wine Quality Dataset:")
display(wine_quality_data.head())
```

Below the code, the first few rows of each dataset are displayed as tables. The Vehicle Dataset table has columns: Car_Name, Year, Selling_Price, Present_Price, Kms_Driven, Fuel_Type, Seller_Type, Transmission, Owner. The Heart Disease Dataset table has columns: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal, condition. The Wine Quality Dataset table has columns: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sulphates, alcohol, quality.

On the right, the Anaconda Assistant chat window shows an error message: "KernelReplyNotOK: SyntaxError (unicode error) 'unicodeescape' codec can't decode bytes in position...". The assistant explains that the error is due to the file path containing backslashes, which are interpreted as escape sequences in Python. It provides two solutions: using double backslashes (\\) or using a raw string (r''). The assistant offers to provide the corrected code.

Figura 5. Carga de datasets e impresión de encabezado por confirmación.

A continuación, deberemos de explorar el contenido e información que contiene cada dataset, en consecuencia, distribuiremos el código en la información de estructura mediante el comando `.info()` y el comando `.describe()` para las estadísticas generales.

Permitiéndonos explorar con mayor precisión la información contenida.

JupyterLab interface showing the execution of code to print information about three datasets: Vehicle, Heart Disease, and Wine Quality. The output displays the structure and statistics for each dataset.

```
[9]: # Estructura
print("Información del Vehicle Dataset:")
vehicle_data.info()

print("Información del Heart Disease Dataset:")
heart_disease_data.info()

print("Información del Wine Quality Dataset:")
wine_quality_data.info()

# Estadísticas
print("Estadísticas del Vehicle Dataset:")
print(vehicle_data.describe())

print("Estadísticas del Heart Disease Dataset:")
print(heart_disease_data.describe())

print("Estadísticas del Wine Quality Dataset:")
print(wine_quality_data.describe())
```

Información del Vehicle Dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 381 entries, 0 to 380
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  --
 0   Car_Name    381 non-null    object
 1   Year        381 non-null    int64
 2   Selling_Price 381 non-null    float64
 3   Present_Price 381 non-null   float64
 4   Kms_Driven  381 non-null    int64
 5   Fuel_Type   381 non-null    object
 6   Seller_Type  381 non-null    object
 7   Transmission 381 non-null    object
 8   Owner       381 non-null    int64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
Información del Heart Disease Dataset:
```

Figura 6. Información de los datasets.

JupyterLab interface showing the execution of code to print statistics for the three datasets. The output displays detailed statistical summaries for each dataset.

```
[9]: # Estructura
print("Información del Vehicle Dataset:")
vehicle_data.info()

print("Información del Heart Disease Dataset:")
heart_disease_data.info()

print("Información del Wine Quality Dataset:")
wine_quality_data.info()

# Estadísticas
print("Estadísticas del Vehicle Dataset:")
print(vehicle_data.describe())

print("Estadísticas del Heart Disease Dataset:")
print(heart_disease_data.describe())

print("Estadísticas del Wine Quality Dataset:")
print(wine_quality_data.describe())
```

10 alcohol 1599 non-null float64
11 quality 1599 non-null int64
dtypes: float64(11), int64(1)
memory usage: 158.0 KB

Estadísticas del Vehicle Dataset:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	381.000000	381.000000	381.000000	381.000000	381.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.002812	8.644115	38896.883882	0.247915
min	2001.000000	0.180000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

Estadísticas del Heart Disease Dataset:

	age	sex	cp	trestbps	chol	fb
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

Figura 7. Estadísticas de los datasets.

6.2 Preprocesamiento de datos.

Esta actividad requiere de múltiples pasos, inicialmente debemos de identificar los valores nulos en nuestras bases, una vez los hemos identificado y adicionamos en estos espacios un promedio que no perjudique nuestra estadística. Debemos de eliminarlos. En este punto ya habríamos realizado la limpieza de los datos, ahora deberemos de tratar los valores faltantes y transformarlos. En consecuencia, el tratamiento y transformado de datos implicara la conversión de variables cualitativas a numéricas como también la disminución de influencia de las variables atípicas por normalización de la información.

6.2.1 Limpieza de datos

Como mencionamos, aquí inicialmente identificamos donde están los valores nulos para cada fila, e imprimimos esta información para que nos sea perceptible.

Sin embargo, los eliminamos y reemplazamos para que estadísticamente nuestros resultados sean mas fiables respecto a la información verídica.

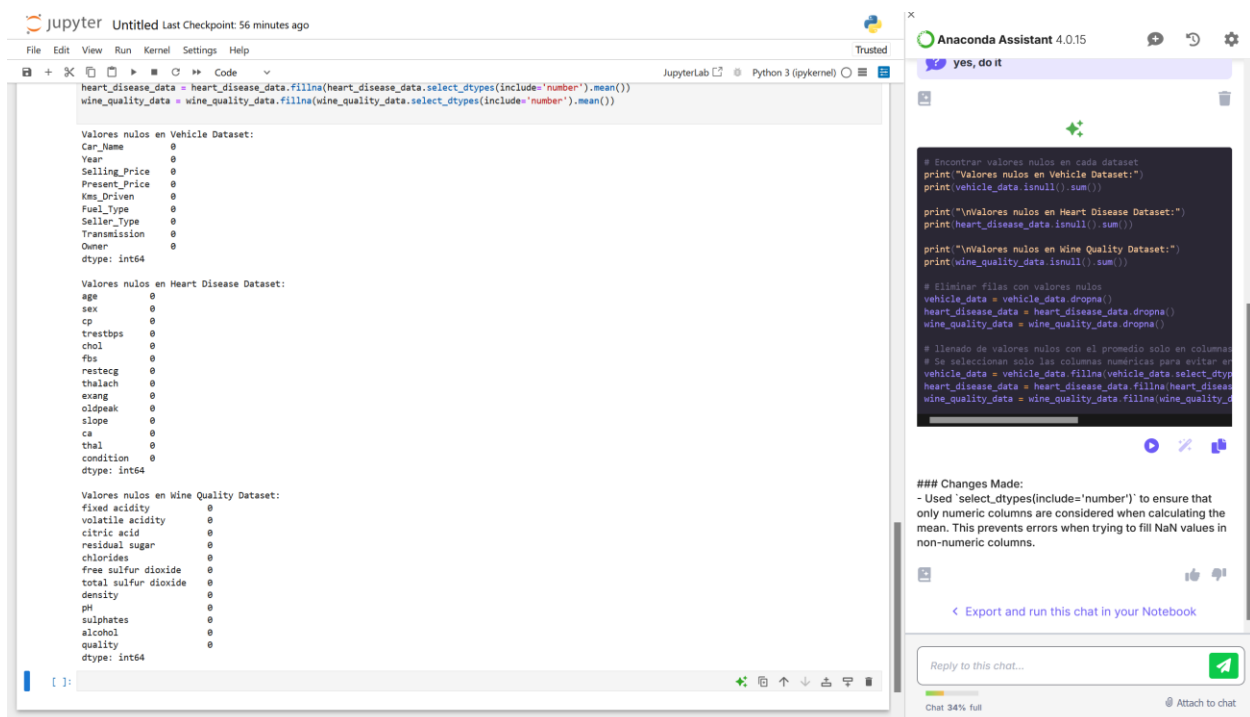


Figura 8. Resultado de investigación de valores nulos (No identificados para el caso).

Durante el desarrollo del proceso de limpieza, hubo problemas con la identificación de información en las columnas que no poseían valores numéricos, en consecuencia, con la ayuda de “Anaconda Assistant” se logro identificar y corregir el código para la única aplicación en celdas numéricas.

6.2.2 Tratamiento de datos

En este punto y por facilidad de interpretación, realizaremos la conversión de variables categóricas a numéricas, como es el caso de las variables “fuel, seller_type, transmission”.

Sin embargo, una vez explorados los datos, podremos comprender que los datasets de hearth diseases y wine quality usan únicamente datos con variables numéricas, en consecuencia, no necesitaremos aplicar el tratamiento de datos sobre estos conjuntos.

The image shows a Jupyter Notebook interface with a code cell and an Anaconda Assistant chat window. The code cell contains the following Python code:

```
[21]: # Cambio de variables categóricas
print(vehicle_data.columns)

columns_to_encode = ['Fuel_Type', 'Seller_Type', 'Transmission', 'Owner']
existing_columns = [col for col in columns_to_encode if col in vehicle_data.columns]

vehicle_data = pd.get_dummies(vehicle_data, columns=existing_columns, drop_first=True)
```

The output of the code cell shows the columns of the 'vehicle_data' DataFrame:

```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type_Diesel', 'Fuel_Type_Petrol', 'Seller_Type_Individual',
       'Transmission_Manual', 'Owner_1', 'Owner_3'],
      dtype='object')
```

The Anaconda Assistant chat window displays an error message:

```
KernelReplyNotOK: KeyError "None of
[Index(['Fuel_Type', 'Seller_Type', 'Transmission',
       'Owner'],...]
```

The chat window provides a detailed explanation of the error and a solution:

The error occurs because the specified columns ('Fuel_Type', 'Seller_Type', 'Transmission', and 'Owner') do not exist in the 'vehicle_data' DataFrame. This could be due to a typo in the column names or because the DataFrame does not contain these columns at all.

To fix this error, you should first check the actual column names in the 'vehicle_data' DataFrame to ensure that they match the names you are trying to use. You can do this by printing the columns of the DataFrame.

Would you like me to provide the corrected code that includes a check for the column names?

At the bottom of the chat window, there is a text input field with the placeholder "Reply to this chat..." and a green button with a paper plane icon. Below the input field, it says "Chat 10% full" and "Attach to chat".

Figura 9. Tratamiento de datos por conversión a variable numérica.

6.2.3 Transformación de datos

Finalmente, debemos de normalizar la distribución de las variables para que los datos tengan mayor correlación. Aun así, para continuar con este proceso es necesario identificar cual es el dataset que efectivamente es sensible a la magnitud de la variable, si identificamos la información contenida en cada algoritmo, comprenderemos que no todos cuentan con esta característica y únicamente deberemos de aplicar este proceso sobre el dataset de vinos de calidad, teniendo en cuenta que sus resultados son muy sensibles a la magnitud de sus variables.

En este orden podemos continuar con el código que responde al caso:

The screenshot displays a JupyterLab environment with a code editor on the left and an Anaconda Assistant chat window on the right. The code in the editor is as follows:

```
[21]: # Cambio de variables categóricas
print(vehicle_data.columns)

columns_to_encode = ['Fuel_Type', 'Seller_Type', 'Transmission', 'Owner']
existing_columns = [col for col in columns_to_encode if col in vehicle_data.columns]

vehicle_data = pd.get_dummies(vehicle_data, columns=existing_columns, drop_first=True)

Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
      'Fuel_Type_Diesel', 'Fuel_Type_Petrol', 'Seller_Type_Individual',
      'Transmission_Manual', 'Owner_1', 'Owner_3'],
      dtype='object')
```

```
[23]: from sklearn.preprocessing import MinMaxScaler

# Creado de contador
scaler = MinMaxScaler()

# Escalar datos numéricos en el Wine Quality dataset
wine_quality_data[wine_quality_data.columns] = scaler.fit_transform(wine_quality_data[wine_quality_data.columns])

# Revisar el dataset escalado
wine_quality_data.head()
```

The output of the code shows the columns of the 'vehicle_data' DataFrame and the first five rows of the 'wine_quality_data' DataFrame:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	0.247788	0.397260	0.00	0.068493	0.106845	0.140845	0.098940	0.567548	0.606299	0.137725	0.153846	0.4
1	0.283186	0.520548	0.00	0.116438	0.143573	0.338028	0.215548	0.494126	0.362205	0.209581	0.215385	0.4
2	0.283186	0.438356	0.04	0.095890	0.133556	0.197183	0.169611	0.508811	0.409449	0.191617	0.215385	0.4
3	0.584071	0.109589	0.56	0.068493	0.105175	0.225352	0.190813	0.582232	0.330709	0.149701	0.215385	0.6
4	0.247788	0.397260	0.00	0.068493	0.106845	0.140845	0.098940	0.567548	0.606299	0.137725	0.153846	0.4

The Anaconda Assistant chat window on the right displays an error message:

```
Debug error using error cell and error output:
KernelReplyNotOK: KeyError "None of
[Index(['Fuel_Type', 'Seller_Type', 'Transmission',
'Owner'],...]
```

The assistant explains the error and provides a solution:

The error occurs because the specified columns ('Fuel_Type', 'Seller_Type', 'Transmission', and 'Owner') do not exist in the 'vehicle_data' DataFrame. This could be due to a typo in the column names or because the DataFrame does not contain these columns at all.

To fix this error, you should first check the actual column names in the 'vehicle_data' DataFrame to ensure that they match the names you are trying to use. You can do this by printing the columns of the DataFrame.

Would you like me to provide the corrected code that includes a check for the column names?

Export and run this chat in your Notebook

Reply to this chat...

Chat 10% full

Attach to chat

Figura 10. Transformación de datos para distribuir correlación entre variables.

6.3 Selección de características relevantes

Debido a que tenemos múltiples datasets, tendremos que realizar el proceso para cada uno de los conjuntos por aparte por celda de código.

6.3.1 Vehicle Dataset

Siguiendo la estructura propuesta, usaremos la regresión lineal donde podemos usar la correlación entre las variables para ver cuáles tienen una relación significativa con la variable objetivo que en este caso es `selling_price`.

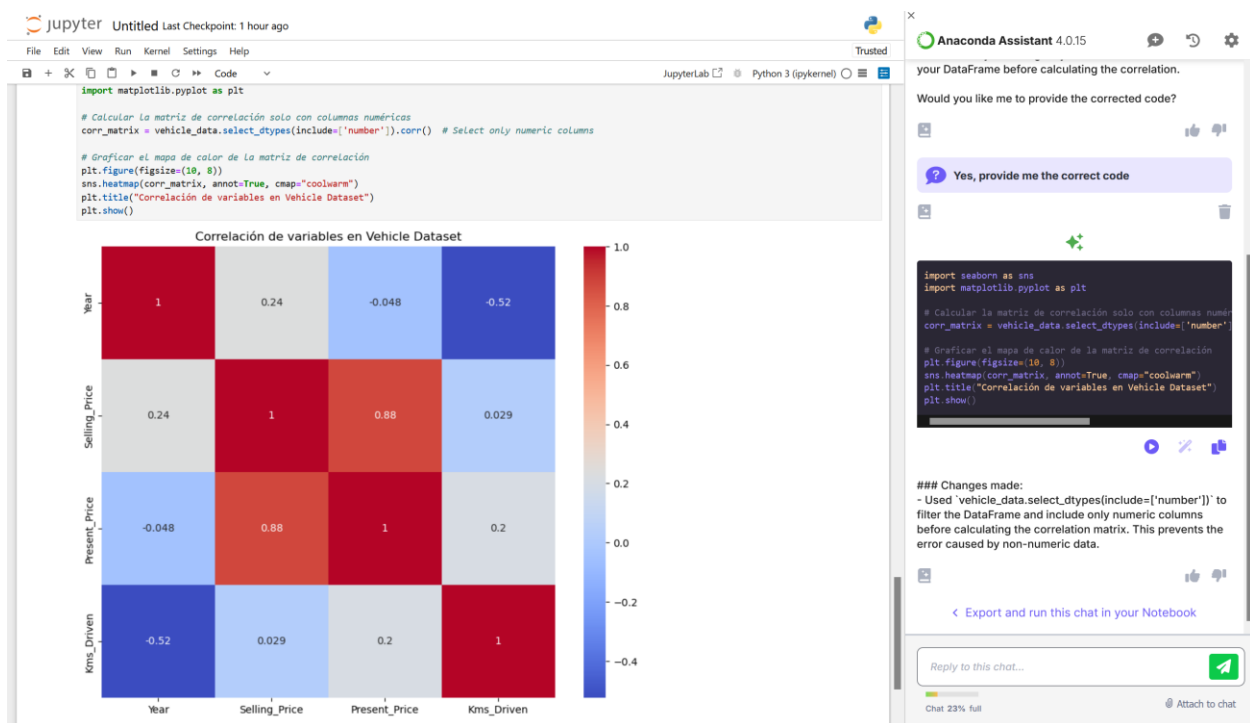


Figura 11. Correlación de variables en el dataset de vehículos.

6.3.2 Heart Diseases Dataset

Ahora implementaremos la información de la base de las defunciones por problemas asociados al corazón, en este orden usaremos la regresión lógica como se presente a continuación:

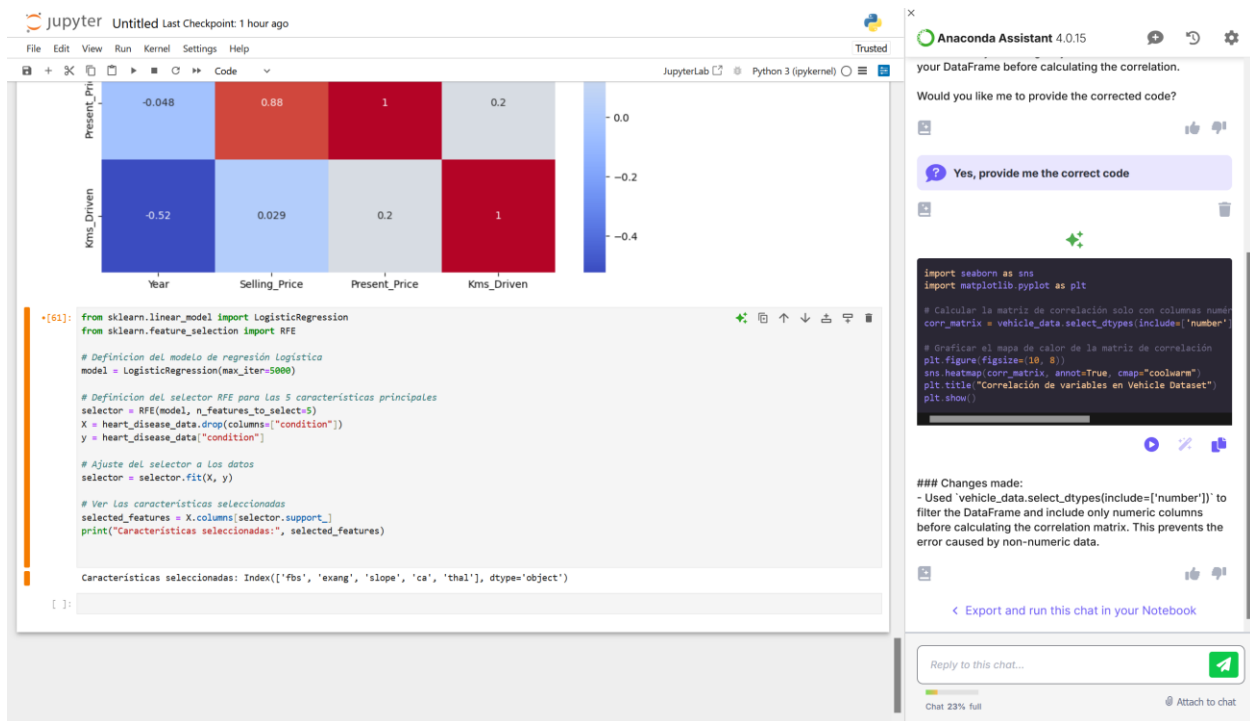


Figura 12. Principales características seleccionadas por regresión logística.

6.3.3 Wine Quality Dataset

Para el dataset de vino, usaremos la importancia de características que nos proporciona el árbol de decisión. Esto nos da una medida de cuánta información aporta cada variable al modelo de clasificación.

Para este caso, al estar evaluando la calidad del vino, la importancia podrá representar como afecta al puntaje final del vino, cada una de las variables que se exponen en la base de datos (Figura 13).

6.4 Division del dataset (Wine Quality)

Para efectos de continuar los procesos, seleccionaremos únicamente el dataset de Wine Quality. Sobre este aplicaremos el proceso para este y los demás puntos que se continúan a partir del presente.

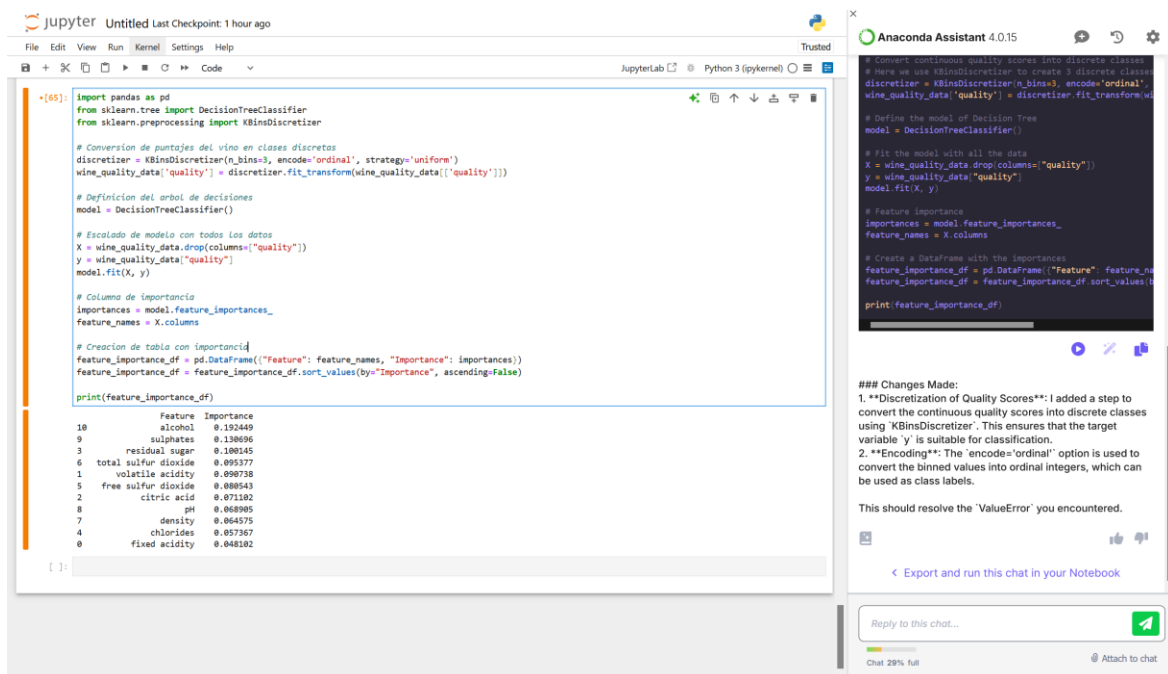


Figura 13. Importancia de variable en la puntuación final del vino.

En este apartado, la división del dataset en entrenamiento y prueba nos permitirá evaluar el rendimiento del modelo en datos que no haya visto antes.

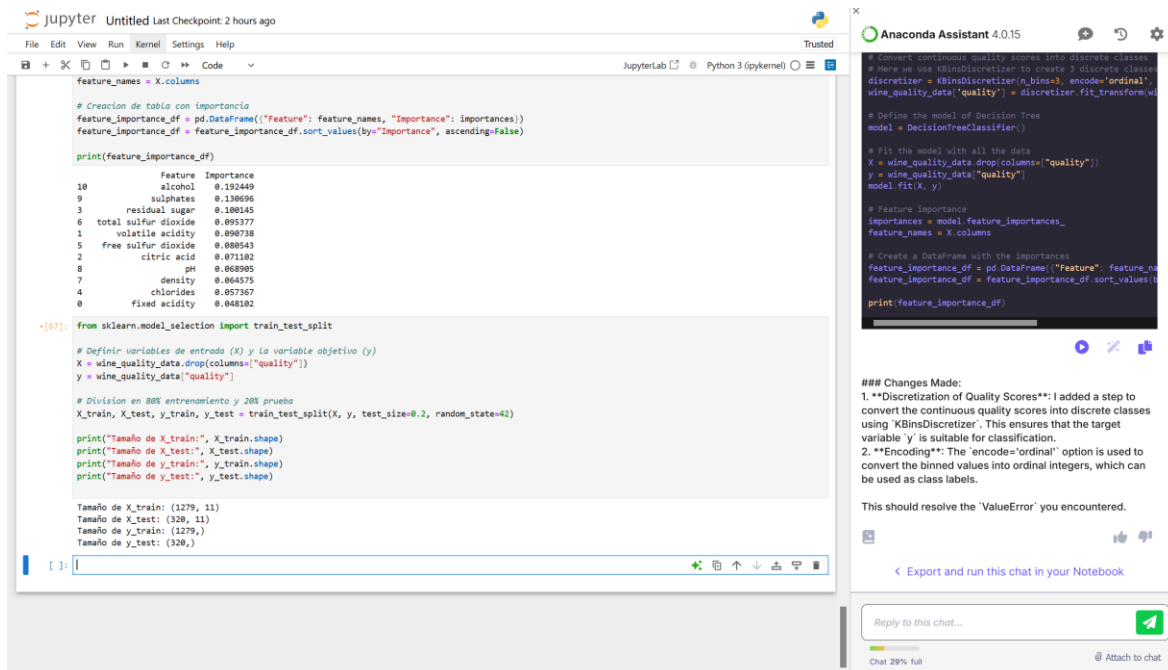
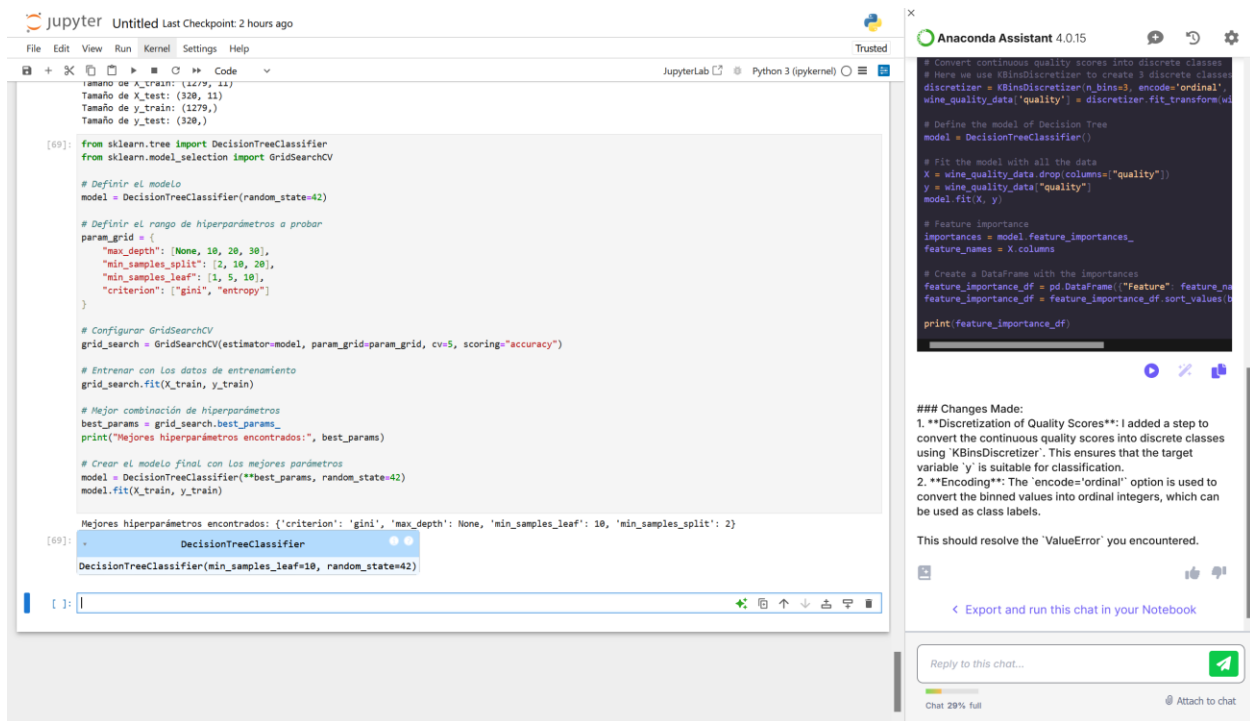


Figura 14. División de Train y Test.

6.5 Entrenamiento de modelo

Ahora se realiza el entrenamiento del modelo usando el árbol de decisión y ajustamos sus hiperparámetros con GridSearchCV para encontrar los valores óptimos, así:



```
[09]: from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

# Definir el modelo
model = DecisionTreeClassifier(random_state=42)

# Definir el rango de hiperparámetros a probar
param_grid = {
    "max_depth": [None, 10, 20, 30],
    "min_samples_split": [2, 10, 20],
    "min_samples_leaf": [1, 5, 10],
    "criterion": ["gini", "entropy"]
}

# Configurar GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring="accuracy")

# Entrenar con los datos de entrenamiento
grid_search.fit(X_train, y_train)

# Mejor combinación de hiperparámetros
best_params = grid_search.best_params_
print("Mejores hiperparámetros encontrados:", best_params)

# Crear el modelo final con los mejores parámetros
model = DecisionTreeClassifier(**best_params, random_state=42)
model.fit(X_train, y_train)

Mejores hiperparámetros encontrados: {'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 10, 'min_samples_split': 2}

[09]: DecisionTreeClassifier
DecisionTreeClassifier(min_samples_leaf=10, random_state=42)
```

Figura 15. Entrenamiento del modelo.

6.6 Evaluación del modelo

Para evaluar el modelo en el conjunto de prueba, calcularemos métricas de precision, recall y F1-score.

Es importante resaltar que estos modelos son evaluados mediante las métricas que fueron explicadas en la definición de los datos, haciendo que este proceso tenga una opción de evaluación estándar y así determinar en condiciones similares los resultados del procedimiento realizado.

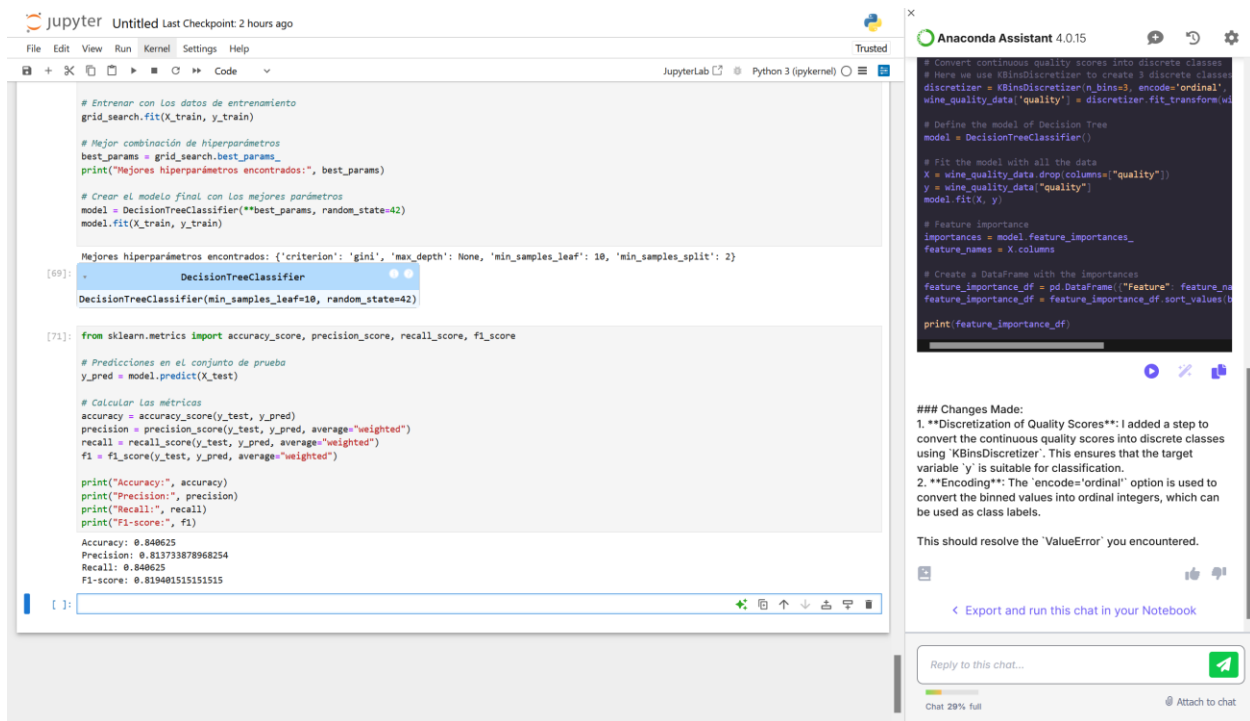


Figura 16. Evaluación del modelo.

6.7 Gráfico del árbol de decisión

Finalmente, para comprender las decisiones a las que el modelo tuvo que identificar para llegar a un resultado, se grafica el árbol de decisión que realizó de manera autónoma para que alcanzara sus resultados (Figura 17).

6.8 Interpretación de resultados

Retomando la información establecida en las definiciones para comprender este desarrollo, sabremos comprender que con las métricas de accuracy, precision, recall y F1-score, podemos saber cómo de bien está funcionando nuestro modelo en términos de predicción de la calidad del vino. Que des englobando cada resultado:

- Accuracy (Precisión global): El modelo acertó aproximadamente el 84.06% de las predicciones en el conjunto de prueba sugiriendo que el modelo es efectivo en la clasificación de la calidad del vino.

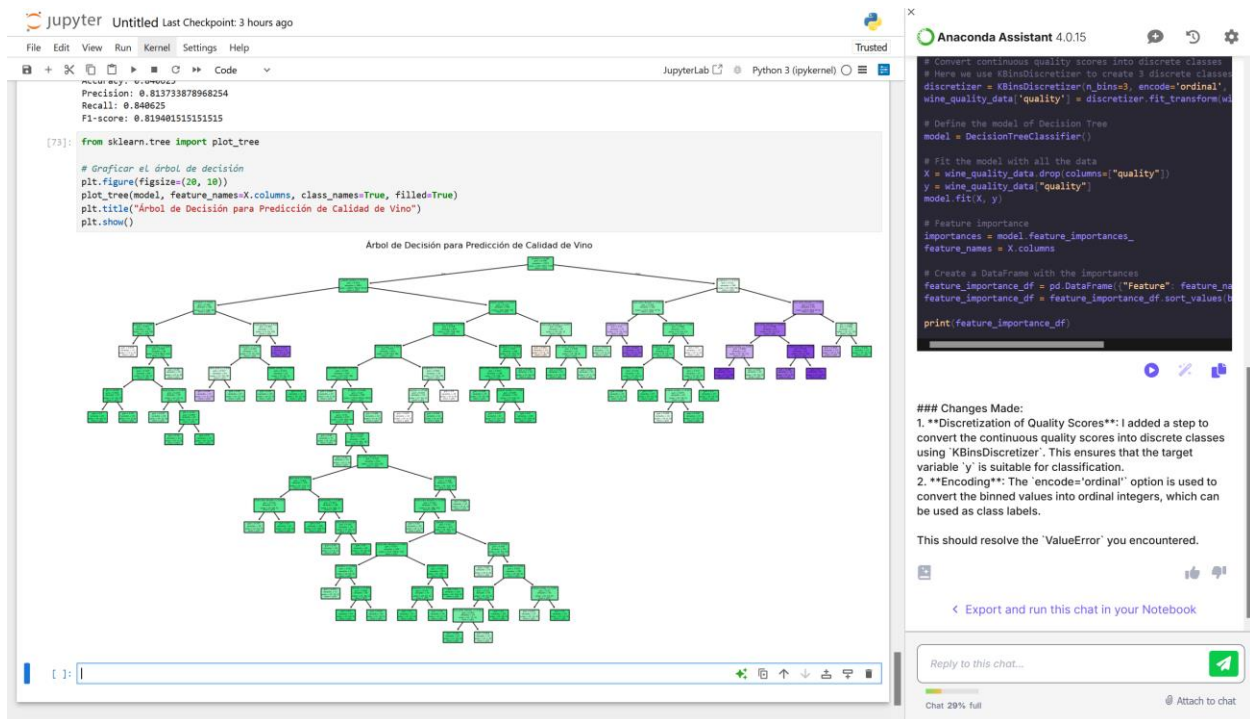


Figura 17. Árbol de decisión.

- **Precisio:** Indica que de todas las veces que el modelo predijo una calidad específica, el 81.37% de esas predicciones fueron correctas, permitiendo que este modelo minimice el riesgo de clasificar un vino como de alta calidad cuando no lo es.
- **Recall (Sensibilidad):** Significa que el modelo fue capaz de identificar correctamente el 84.06% de las instancias de calidad específica en el conjunto de prueba.
- **F1-Score:** Es la media armónica de la precisión y el recall. Este indicador es útil cuando identificar si el un conjunto de datos es desbalanceado, ya que combina ambas métricas en una sola. En este caso al estar cercano a uno (1) indica que el modelo tiene un buen equilibrio entre precisión y recall.

Una vez reunida esta información, se determina que el modelo de árbol de decisión ha tiene un rendimiento sólido en la clasificación de la calidad del vino. Además, las métricas indican que es capaz de predecir con precisión al momento de la toma de decisiones.

7 BIBLIOGRAFÍA

Arce, J. I. B. (2019, julio 26). La matriz de confusión y sus métricas. Juan Barrios; Juan Ignacio Barrios Arce. <https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/>

Brownlee, J. (2020, agosto 14). What is the Difference Between Test and Validation Datasets? Machinelearningmastery.com. <https://machinelearningmastery.com/difference-test-validation-datasets/>

de los Santos, P. R. (2022, enero 24). Datos de entrenamiento vs datos de test. Telefónica Tech. <https://telefonicatech.com/blog/datos-entrenamiento-vs-datos-de-test>

GeeksforGeeks. (2019, junio 12). One hot encoding in machine learning. GeeksforGeeks. <https://www.geeksforgeeks.org/ml-one-hot-encoding/>

scikit. (s. f.). GridSearchCV. Scikit-Learn. Recuperado 27 de octubre de 2024, de https://scikit-learn.org/dev/modules/generated/sklearn.model_selection.GridSearchCV.html

8 ANOTACIÓN DE ENTREGA

Tutor, disculpe usted esta actividad suponía ser desarrollo grupal, pero la realice individual debido a que lamentablemente mi equipo tuvo un daño en el cargador y mi viejo computador me genero muchos problemas de rendimiento para cumplir con mis deberes teniendo en cuenta que no era capaz de manejar los procesos por especificaciones técnicas de este.

Le deseo un gran mes.

David Esteban Lasso O.