

MANUAL DE USUARIOS: MODELO DE MACHINE LEARNING PARA IDENTIFICAR POTENCIALES CLIENTES DE SEGUROS DE AUTOMÓVILES

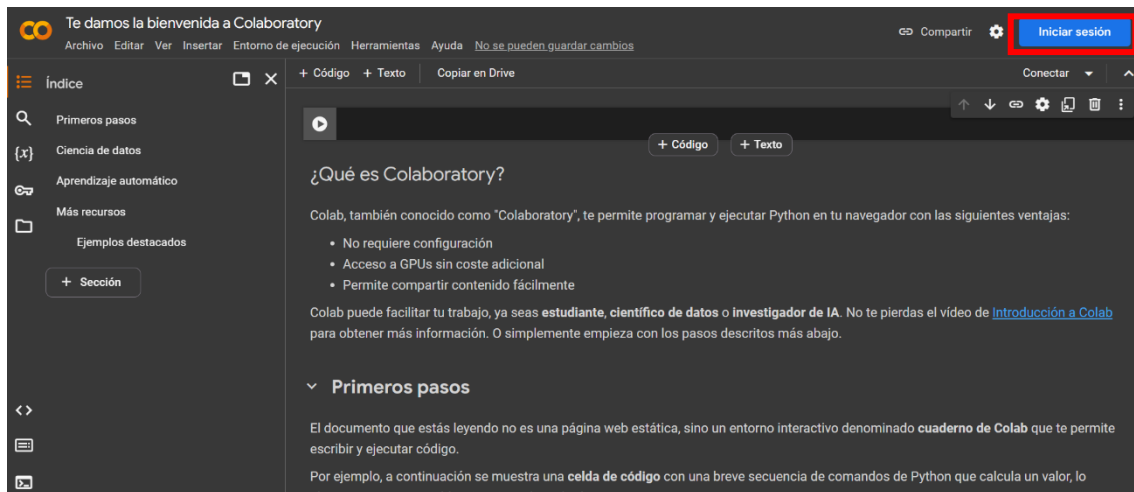
Requerimientos mínimos para la ejecución del modelo:

- Contar con una cuenta Google
- Tener instalado Power Bi desktop en el equipo de prueba
- Descargar los archivos de la ruta:

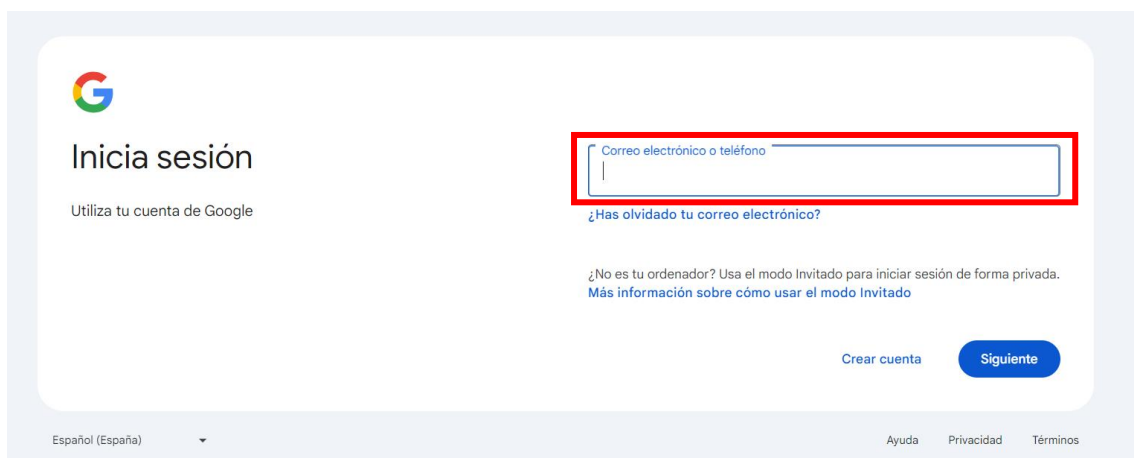
https://drive.google.com/drive/folders/17Bna9hookl_u0apsDV2ml8iINpGBHcU6?usp=sharing

Paso 1: Login en Google Colab (CP001)

Dar click en iniciar sesión



Ingresar tu correo electrónico



Ingresar tu contraseña



Te damos la bienvenida

 alexandermedinacortez@gmail.com

Mostrar contraseña

[¿Has olvidado tu contraseña?](#) [Siguiente](#)

Español (España) Ayuda Privacidad Términos

Sesión Exitosa

Te damos la bienvenida a Colaboratory

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

+ Código + Texto Copiar en Drive

Conectar Colab AI

Índice

Primeros pasos

Ciencia de datos

Aprendizaje automático

Más recursos

Ejemplos destacados


+ Sección

Te damos la bienvenida a Colab

(Novedad) Prueba la API de Gemini

- Generate a Gemini API key
- Talk to Gemini with the Speech-to-Text API
- Gemini API Quickstart with Python
- Gemini API code sample
- Compare Gemini with ChatGPT
- More notebooks

Si ya conoces Colab, echa un vistazo a este vídeo para obtener información sobre las tablas interactivas, la vista del historial de código ejecutado y la paleta de comandos.



[] Explora a programar o a crear código con IA.

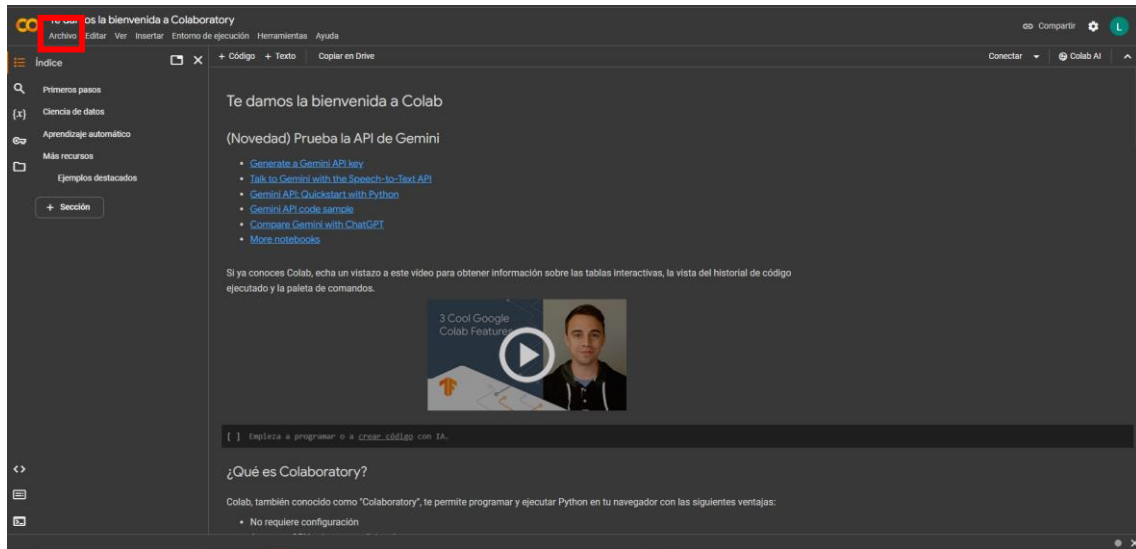
¿Qué es Colaboratory?

Colab, también conocido como "Colaboratory", te permite programar y ejecutar Python en tu navegador con las siguientes ventajas:

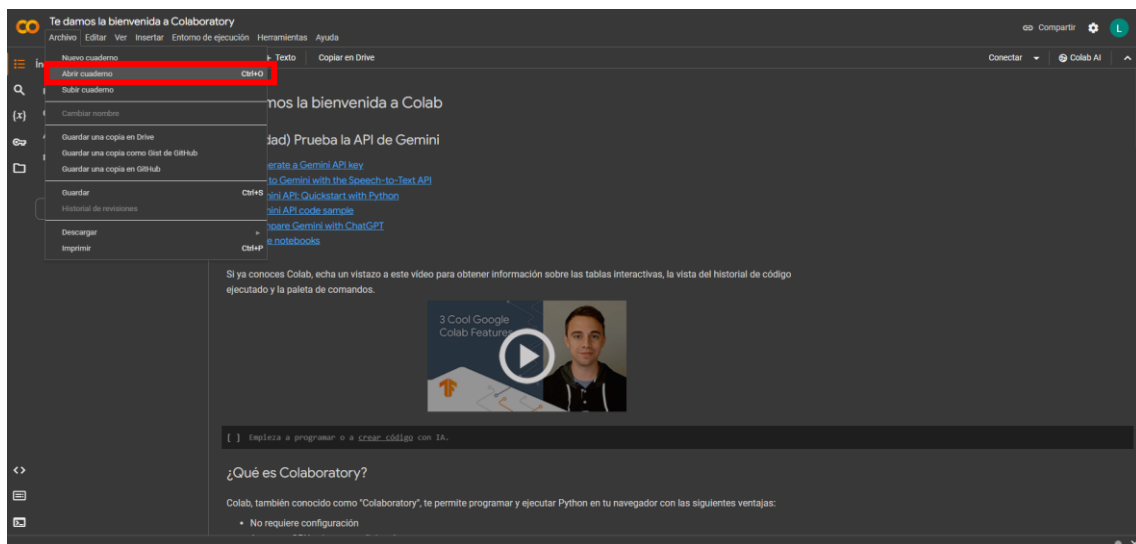
- No requiere configuración.

Paso 2: Cargar el algoritmo a Google Colab (CP002)

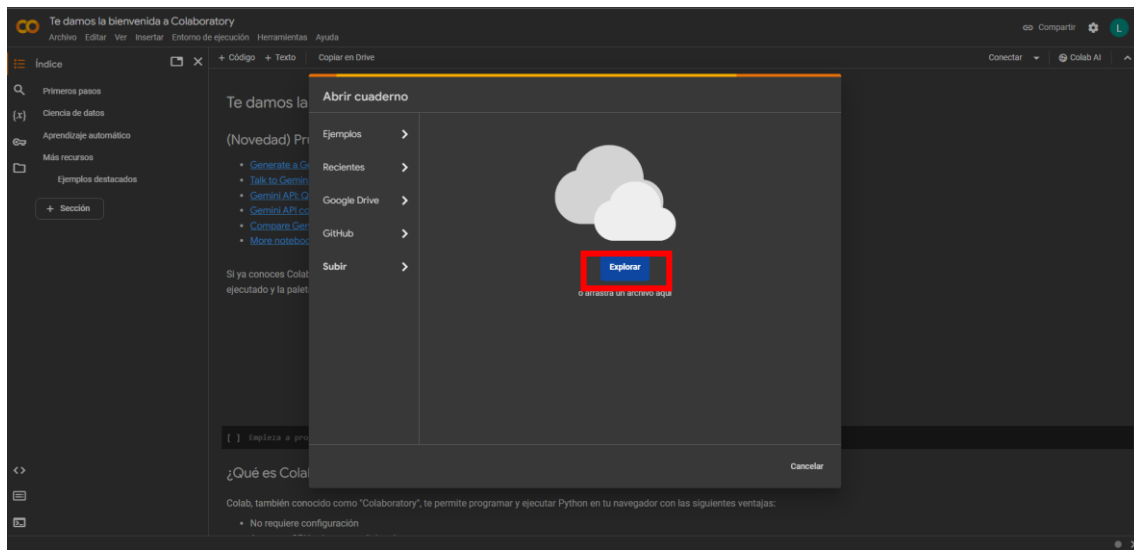
Dar click en “Archivo”



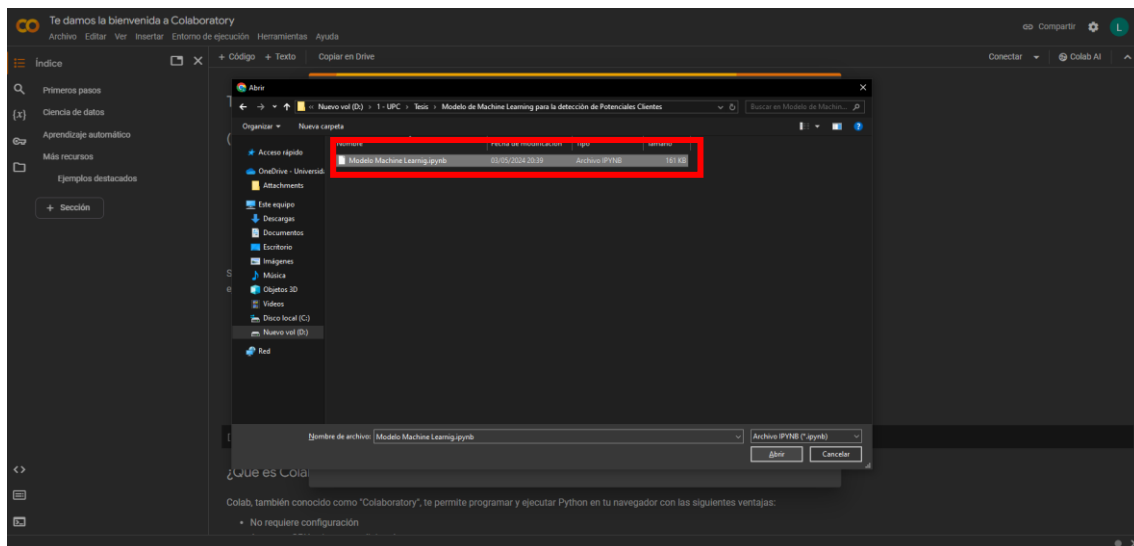
Dar click en “Abrir cuaderno”



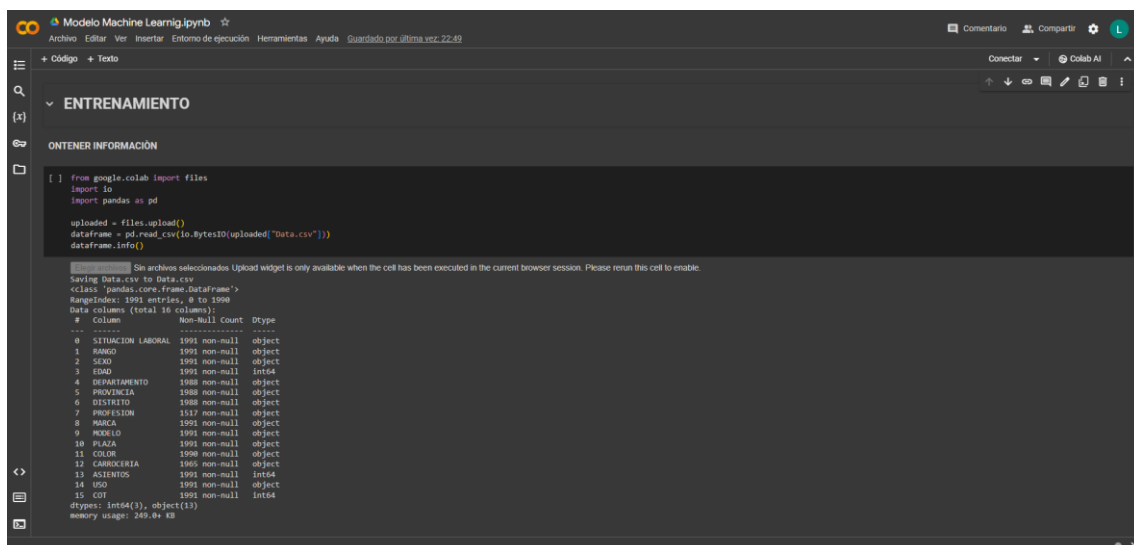
Dar click en “Subir” > “Explorar”



Subes el archivo “Modelo Machine Learning.ipynb”

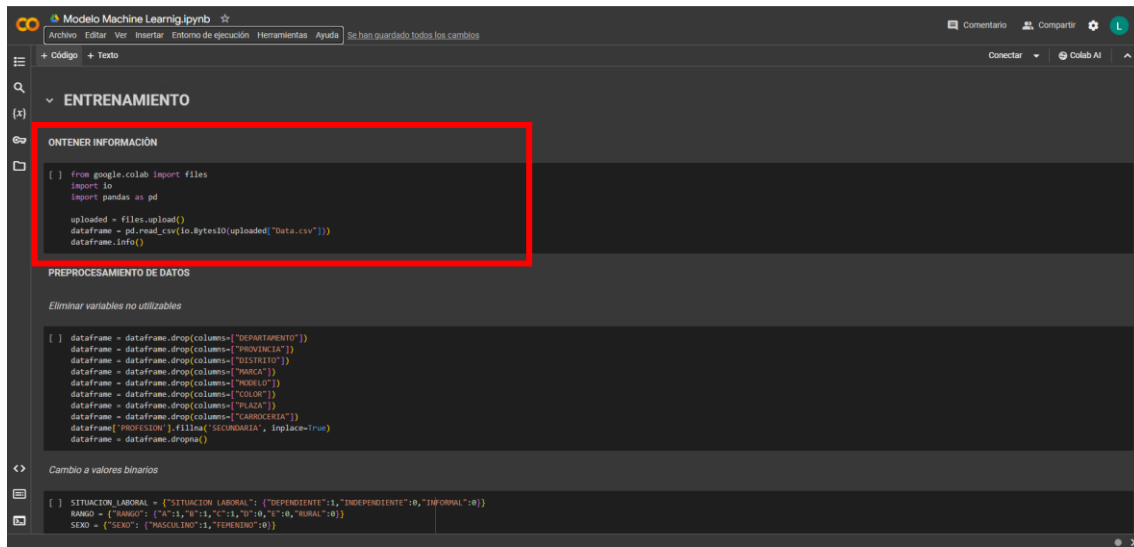


Carga exitosa



Paso 3: Cargar el algoritmo dataset (CP003)

Ejecutar el bloque “Obtener Información” dando click en el botón de play

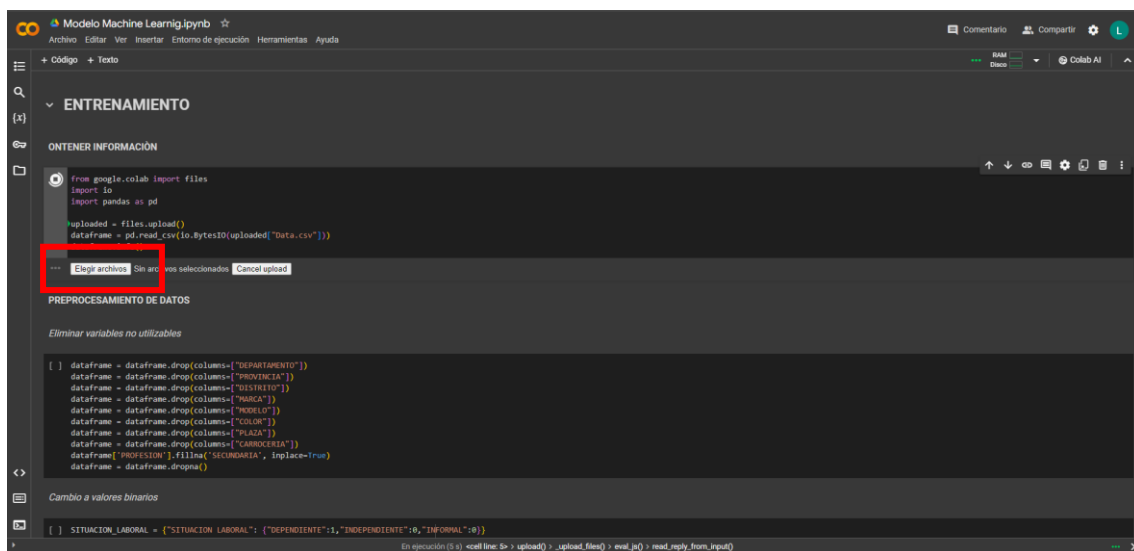


The screenshot shows a Jupyter Notebook titled 'Modelo Machine Learning.ipynb'. The interface includes a top menu bar with options like 'Archivo', 'Editar', 'Ver', 'Insertar', 'Entorno de ejecución', 'Herramientas', and 'Ayuda'. On the left, there's a sidebar with icons for file management and search. The main area is divided into sections: 'ENTRENAMIENTO', 'OBTENER INFORMACIÓN', 'PREPROCESAMIENTO DE DATOS', and 'Cambio a valores binarios'. The 'OBTENER INFORMACIÓN' section is highlighted with a red box and contains the following code:

```
[ ] from google.colab import files
import io
import pandas as pd

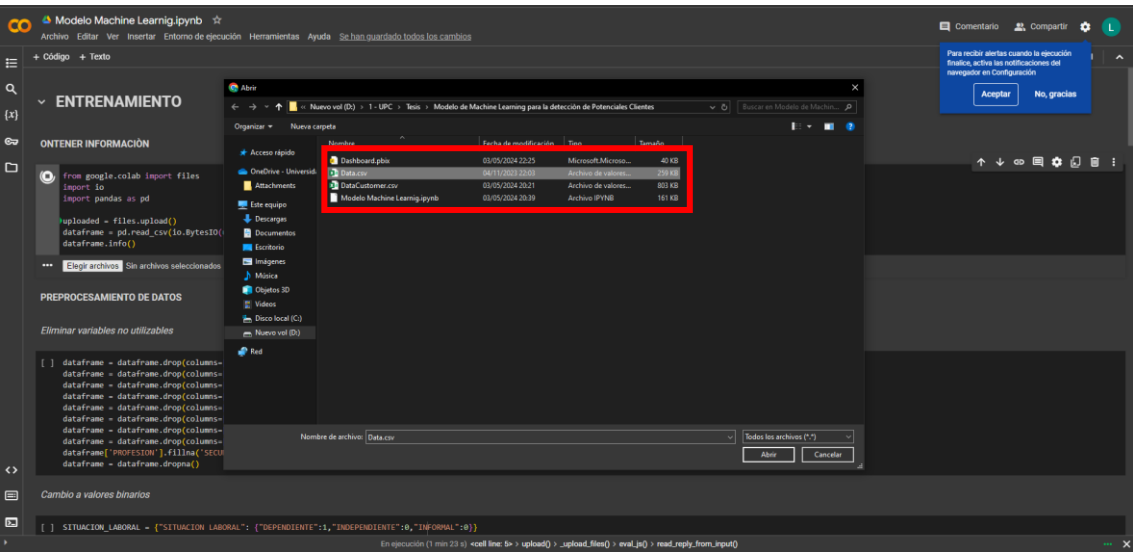
uploaded = files.upload()
dataframe = pd.read_csv(io.BytesIO(uploaded["Data.csv"]))
dataframe.info()
```

Dar click en “Elegir Archivo”

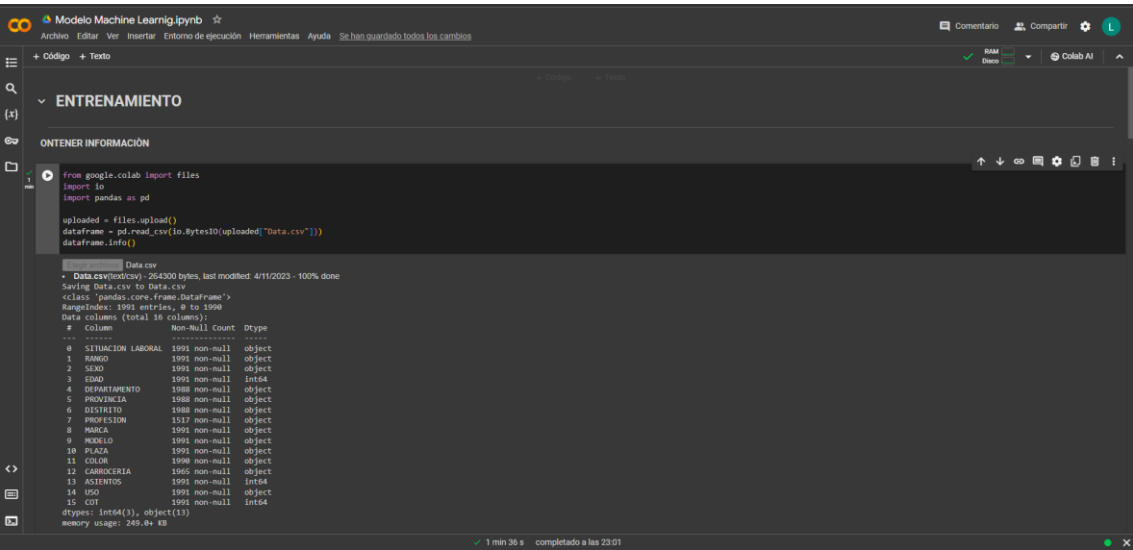


This screenshot shows the same Jupyter Notebook interface, but with the 'OBTENER INFORMACIÓN' section expanded. Below the code block, there is a file upload interface. A red box highlights the 'Elegir archivos' button, which is labeled 'Sin archivos seleccionados' and 'Cancel upload'. The rest of the notebook content, including the 'PREPROCESAMIENTO DE DATOS' and 'Cambio a valores binarios' sections, remains visible below.

Seleccionar el archivo “Data.csv” y click en abrir

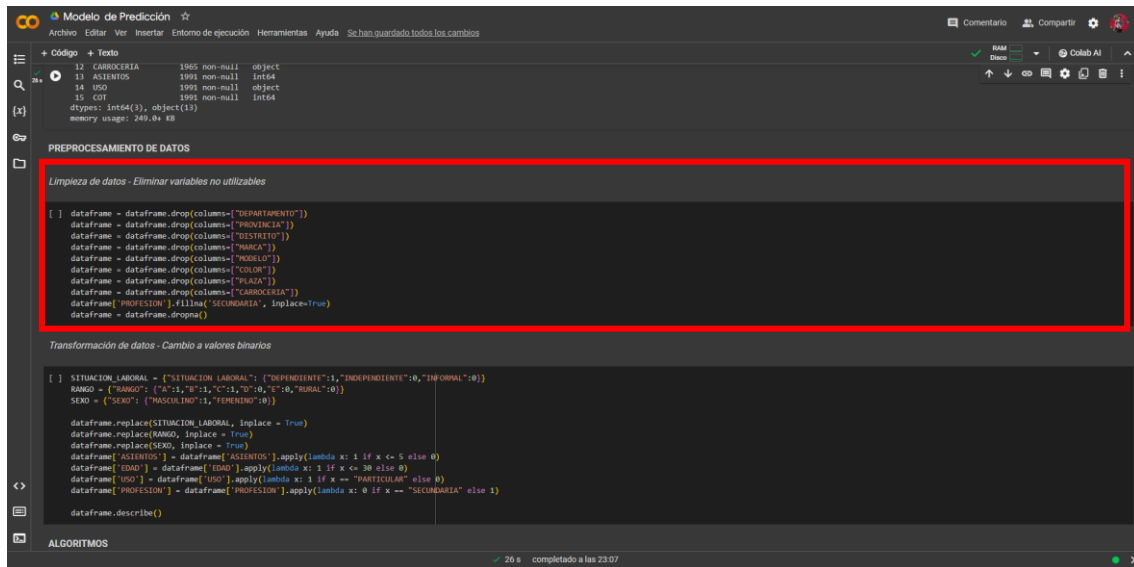


Carga exitosa



Paso 4: Ejecución del algoritmo de limpieza de datos (CP004)

Ejecutar bloque “Limpieza de datos - Eliminar variables no utilizables”



```
12 CARROCERIA 1965 non-null object
13 ASIENTOS 1991 non-null int64
14 USO 1991 non-null object
15 COT 1991 non-null int64
dtypes: int64(3), object(13)
memory usage: 249.0+ KB

PREPROCESAMIENTO DE DATOS

Limpieza de datos - Eliminar variables no utilizables

[ ] dataframe = dataframe.drop(columns=["DEPARTAMENTO"])
dataframe = dataframe.drop(columns=["PROVINCIA"])
dataframe = dataframe.drop(columns=["DISTRITO"])
dataframe = dataframe.drop(columns=["MARCA"])
dataframe = dataframe.drop(columns=["MODELO"])
dataframe = dataframe.drop(columns=["COLOR"])
dataframe = dataframe.drop(columns=["PLAZA"])
dataframe = dataframe.drop(columns=["CARROCERIA"])
dataframe["PROFESION"].fillna("SECUNDARIA", inplace=True)
dataframe = dataframe.dropna()

Transformación de datos - Cambio a valores binarios

[ ] SITUACION_LABORAL = {"SITUACION_LABORAL": {"DEPENDIENTE":1,"INDEPENDIENTE":0,"INFORMAL":0}}
RANGO = {"RANGO": {"A":1,"B":1,"C":1,"D":0,"E":0,"RURAL":0}}
SEXO = {"SEXO": {"MASCULINO":1,"FEMENINO":0}}

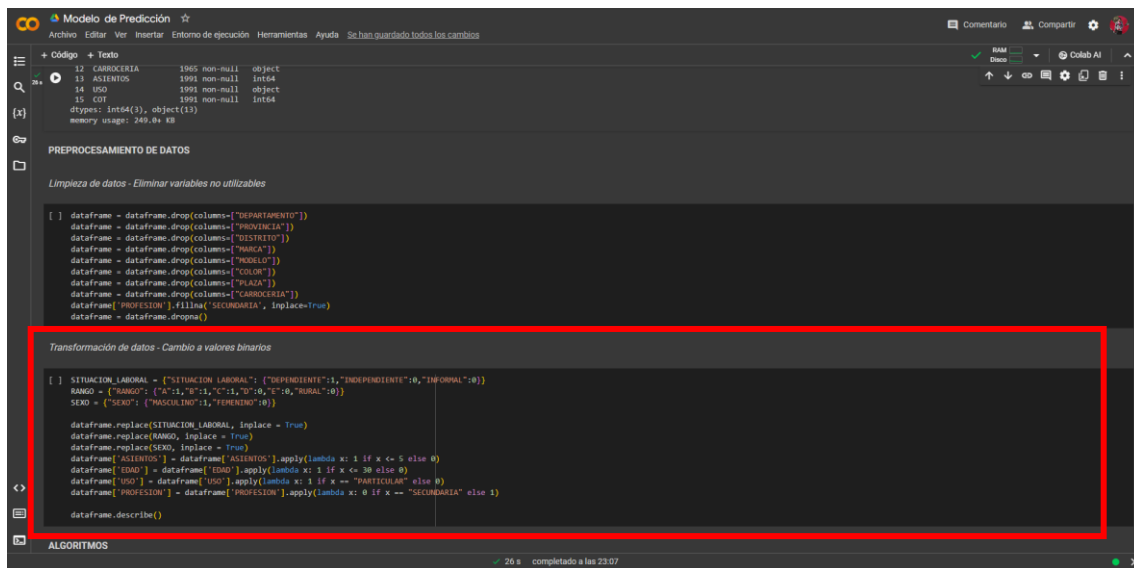
dataframe.replace(SITUACION_LABORAL, inplace = True)
dataframe.replace(RANGO, inplace = True)
dataframe.replace(SEXO, inplace = True)
dataframe["ASIENTOS"] = dataframe["ASIENTOS"].apply(lambda x: 1 if x <= 5 else 0)
dataframe["EDAD"] = dataframe["EDAD"].apply(lambda x: 1 if x <= 30 else 0)
dataframe["USO"] = dataframe["USO"].apply(lambda x: 1 if x == "PARTICULAR" else 0)
dataframe["PROFESION"] = dataframe["PROFESION"].apply(lambda x: 0 if x == "SECUNDARIA" else 1)

dataframe.describe()

ALGORITMOS
26 s completado a las 23:07
```

Paso 5: Ejecución del algoritmo de Transformación de datos (CP005)

Ejecutar bloque “Transformación de datos - Cambio a valores binarios”



```
12 CARROCERIA 1965 non-null object
13 ASIENTOS 1991 non-null int64
14 USO 1991 non-null object
15 COT 1991 non-null int64
dtypes: int64(3), object(13)
memory usage: 249.0+ KB

PREPROCESAMIENTO DE DATOS

Limpieza de datos - Eliminar variables no utilizables

[ ] dataframe = dataframe.drop(columns=["DEPARTAMENTO"])
dataframe = dataframe.drop(columns=["PROVINCIA"])
dataframe = dataframe.drop(columns=["DISTRITO"])
dataframe = dataframe.drop(columns=["MARCA"])
dataframe = dataframe.drop(columns=["MODELO"])
dataframe = dataframe.drop(columns=["COLOR"])
dataframe = dataframe.drop(columns=["PLAZA"])
dataframe = dataframe.drop(columns=["CARROCERIA"])
dataframe["PROFESION"].fillna("SECUNDARIA", inplace=True)
dataframe = dataframe.dropna()

Transformación de datos - Cambio a valores binarios

[ ] SITUACION_LABORAL = {"SITUACION_LABORAL": {"DEPENDIENTE":1,"INDEPENDIENTE":0,"INFORMAL":0}}
RANGO = {"RANGO": {"A":1,"B":1,"C":1,"D":0,"E":0,"RURAL":0}}
SEXO = {"SEXO": {"MASCULINO":1,"FEMENINO":0}}

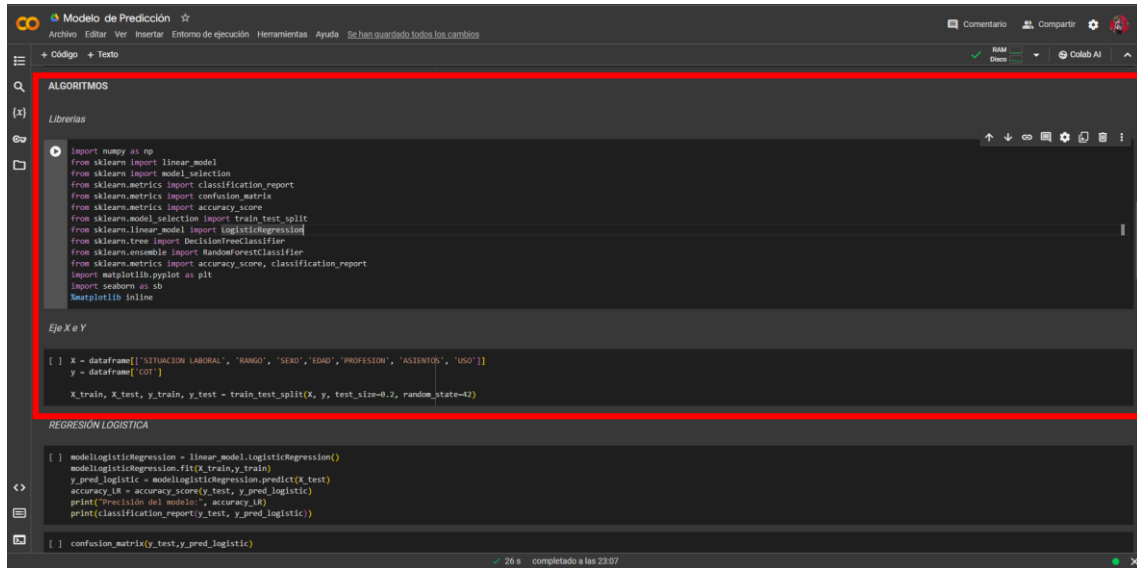
dataframe.replace(SITUACION_LABORAL, inplace = True)
dataframe.replace(RANGO, inplace = True)
dataframe.replace(SEXO, inplace = True)
dataframe["ASIENTOS"] = dataframe["ASIENTOS"].apply(lambda x: 1 if x <= 5 else 0)
dataframe["EDAD"] = dataframe["EDAD"].apply(lambda x: 1 if x <= 30 else 0)
dataframe["USO"] = dataframe["USO"].apply(lambda x: 1 if x == "PARTICULAR" else 0)
dataframe["PROFESION"] = dataframe["PROFESION"].apply(lambda x: 0 if x == "SECUNDARIA" else 1)

dataframe.describe()

ALGORITMOS
26 s completado a las 23:07
```

Paso 6: Ejecución del algoritmo de Regresión Logística (CP006)

Ejecutar los bloques de “Librerías” y “Eje X e Y”



```
import numpy as np
from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sb
%matplotlib inline

# Eje X e Y

X = dataframe[['SITUACION LABORAL', 'RANGO', 'SEXO', 'EDAD', 'PROFESION', 'ASIENTOS', 'USO']]
y = dataframe['COT']

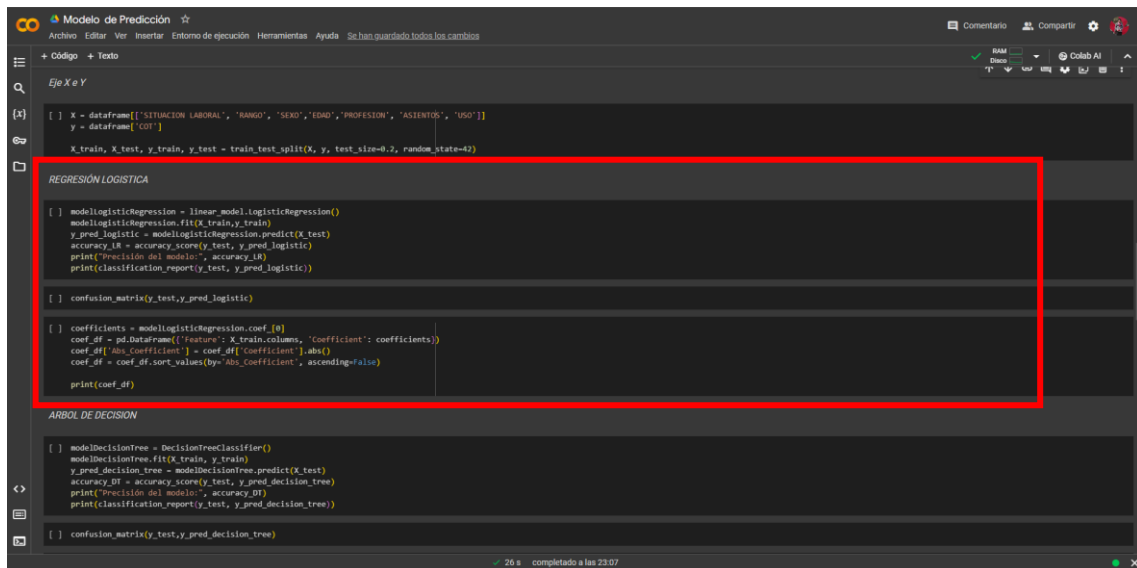
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# REGRESIÓN LOGÍSTICA

modelLogisticRegression = linear_model.LogisticRegression()
modelLogisticRegression.fit(X_train, y_train)
y_pred_logistic = modelLogisticRegression.predict(X_test)
accuracy_LR = accuracy_score(y_test, y_pred_logistic)
print("Precisión del modelo", accuracy_LR)
print(classification_report(y_test, y_pred_logistic))

[ ] confusion_matrix(y_test, y_pred_logistic)
```

Ejecutar el bloque de “Regresión Logística”



```
# REGRESIÓN LOGÍSTICA

modelLogisticRegression = linear_model.LogisticRegression()
modelLogisticRegression.fit(X_train, y_train)
y_pred_logistic = modelLogisticRegression.predict(X_test)
accuracy_LR = accuracy_score(y_test, y_pred_logistic)
print("Precisión del modelo", accuracy_LR)
print(classification_report(y_test, y_pred_logistic))

[ ] confusion_matrix(y_test, y_pred_logistic)

[ ] coefficients = modelLogisticRegression.coef_[0]
coef_df = pd.DataFrame({'feature': X_train.columns, 'coefficient': coefficients})
coef_df['Abs_Coefficient'] = coef_df['coefficient'].abs()
coef_df = coef_df.sort_values(by='Abs_Coefficient', ascending=False)
print(coef_df)
```

```
# ARBOL DE DECISION

modelDecisionTree = DecisionTreeClassifier()
modelDecisionTree.fit(X_train, y_train)
y_pred_decision_tree = modelDecisionTree.predict(X_test)
accuracy_DT = accuracy_score(y_test, y_pred_decision_tree)
print("Precisión del modelo", accuracy_DT)
print(classification_report(y_test, y_pred_decision_tree))

[ ] confusion_matrix(y_test, y_pred_decision_tree)
```


Ejecución Exitosa

```
[6] modelLogisticRegression = linear_model.LogisticRegression()
modelLogisticRegression.fit(X_train,y_train)
y_pred_logistic = modelLogisticRegression.predict(X_test)
accuracy_LR = accuracy_score(y_test, y_pred_logistic)
print("Precisión del modelo:", accuracy_LR)
print(classification_report(y_test, y_pred_logistic))
```

Precisión del modelo: 0.7619047619047619

	precision	recall	f1-score	support
0	0.79	0.65	0.71	182
1	0.74	0.86	0.80	227
accuracy			0.76	399
macro avg	0.77	0.75	0.75	399
weighted avg	0.77	0.76	0.76	399

```
[7] confusion_matrix(y_test,y_pred_logistic)
```

```
array([[118, 64],
       [ 31, 186]])
```

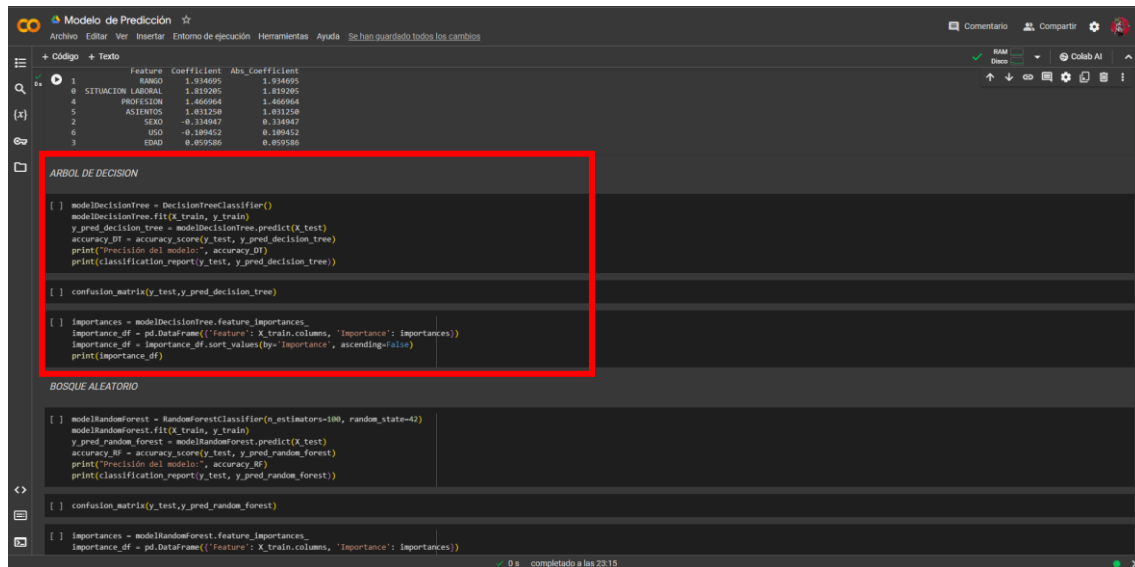
```
[8] coefficients = modelLogisticRegression.coef_[0]
coef_df = pd.DataFrame({'feature': X_train.columns, 'Coefficient': coefficients})
coef_df['Abs_Coefficient'] = coef_df['Coefficient'].abs()
coef_df = coef_df.sort_values(by='Abs_Coefficient', ascending=False)
print(coef_df)
```

	Feature	Coefficient	Abs_Coefficient
1	RANGO	1.934695	1.934695
0	SITUACION LABORAL	1.819285	1.819285
4	PROFESION	1.466964	1.466964
5	ASIENTOS	1.831258	1.831258
2	SEXO	-0.136647	0.136647
6	USO	-0.189452	0.189452
3	EDAD	0.859586	0.859586

0 s completado a las 23:15

Paso 7: Ejecución del algoritmo de Árbol de Decisión (CP007)

Ejecutar el bloque “Árbol de Decisión”



The screenshot shows a Jupyter Notebook interface with a table of features and coefficients, and a code cell for a Decision Tree model. The code is highlighted with a red box.

	Feature	Coefficient	Abs_Coefficient
1	RANGO	1.934695	1.934695
0	SITUACION LABORAL	1.819295	1.819295
4	PROFESION	1.466964	1.466964
5	ASIENTOS	1.831258	1.831258
2	SEXO	-0.334047	0.334047
6	USO	-0.109452	0.109452
3	EDAD	0.859586	0.859586

```
[ ] modelDecisionTree = DecisionTreeClassifier()
modelDecisionTree.fit(X_train, y_train)
y_pred_decision_tree = modelDecisionTree.predict(X_test)
accuracy_DT = accuracy_score(y_test, y_pred_decision_tree)
print("Precisión del modelo:", accuracy_DT)
print(classification_report(y_test, y_pred_decision_tree))

[ ] confusion_matrix(y_test, y_pred_decision_tree)

[ ] importances = modelDecisionTree.feature_importances_
importance_df = pd.DataFrame({'feature': X_train.columns, 'importance': importances})
importance_df = importance_df.sort_values(by='importance', ascending=False)
print(importance_df)

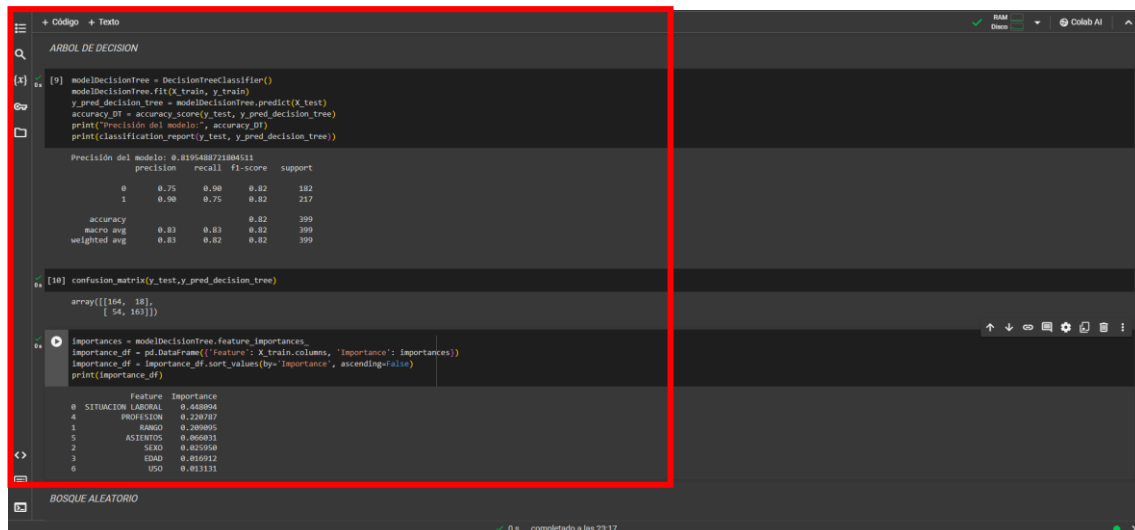
BOSQUE ALEATORIO

[ ] modelRandomForest = RandomForestClassifier(n_estimators=100, random_state=42)
modelRandomForest.fit(X_train, y_train)
y_pred_random_forest = modelRandomForest.predict(X_test)
accuracy_RF = accuracy_score(y_test, y_pred_random_forest)
print("Precisión del modelo:", accuracy_RF)
print(classification_report(y_test, y_pred_random_forest))

[ ] confusion_matrix(y_test, y_pred_random_forest)

[ ] importances = modelRandomForest.feature_importances_
importance_df = pd.DataFrame({'feature': X_train.columns, 'importance': importances})
```

Ejecución Exitosa



The screenshot shows the same Jupyter Notebook interface, but now displaying the output of the code. The output is highlighted with a red box.

```
[9] modelDecisionTree = DecisionTreeClassifier()
modelDecisionTree.fit(X_train, y_train)
y_pred_decision_tree = modelDecisionTree.predict(X_test)
accuracy_DT = accuracy_score(y_test, y_pred_decision_tree)
print("Precisión del modelo:", accuracy_DT)
print(classification_report(y_test, y_pred_decision_tree))

Precisión del modelo: 0.8195488721884511
precision    recall  f1-score   support

0       0.75      0.98      0.82      182
1       0.98      0.75      0.82      217

accuracy          0.83
macro avg         0.83
weighted avg      0.83
```

```
[10] confusion_matrix(y_test, y_pred_decision_tree)

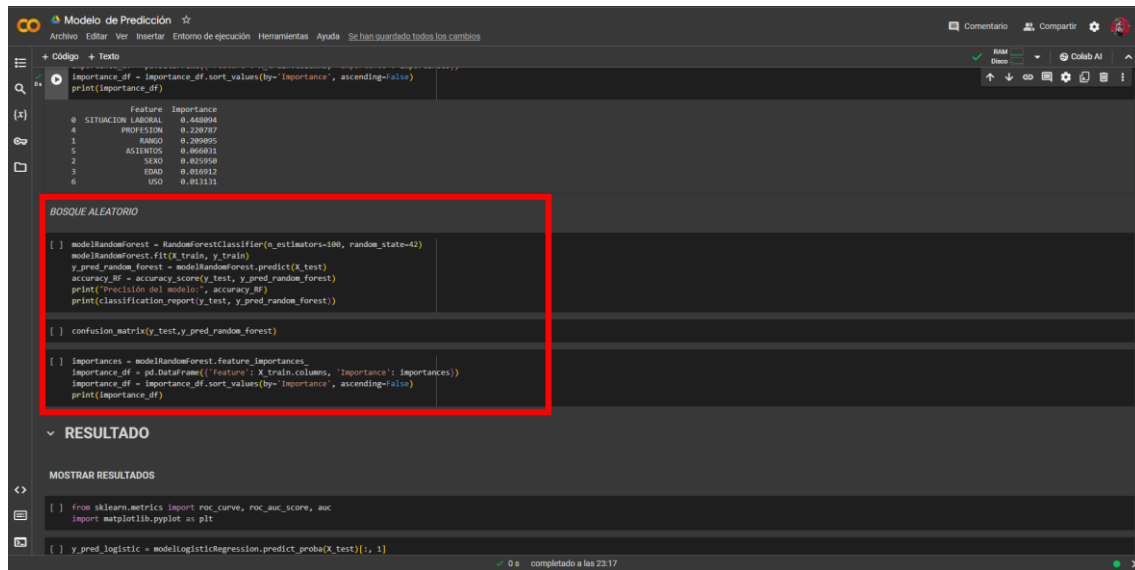
array([[164, 18],
       [ 54, 163]])

[ ] importances = modelDecisionTree.feature_importances_
importance_df = pd.DataFrame({'feature': X_train.columns, 'importance': importances})
importance_df = importance_df.sort_values(by='importance', ascending=False)
print(importance_df)

Feature Importance
0 SITUACION LABORAL 0.468804
4 PROFESION 0.228787
1 RANGO 0.209095
5 ASIENTOS 0.466811
2 SEXO 0.825958
3 EDAD 0.816912
6 USO 0.813131
```

Paso 8: Ejecución del algoritmo de Bosque Aleatorio (CP008)

Ejecutar el bloque “Boque Aleatorio”



```
Importance_df = importance_df.sort_values(by='Importance', ascending=False)
print(importance_df)

Feature Importance
0 SITUACION LABORAL 0.468064
4 PROFESION 0.228787
1 RANGO 0.209895
5 ASIENTOS 0.066931
2 SEXO 0.025958
3 EDAD 0.016912
6 USO 0.013131

BOSQUE ALEATORIO

[ ] modelRandomForest = RandomForestClassifier(n_estimators=100, random_state=42)
modelRandomForest.fit(X_train, y_train)
y_pred_random_forest = modelRandomForest.predict(X_test)
accuracy_RF = accuracy_score(y_test, y_pred_random_forest)
print("Precisión del modelo:", accuracy_RF)
print(classification_report(y_test, y_pred_random_forest))

[ ] confusion_matrix(y_test, y_pred_random_forest)

Importances = modelRandomForest.feature_importances_
importance_df = pd.DataFrame({'feature': X_train.columns, 'importance': importances})
importance_df = importance_df.sort_values(by='Importance', ascending=False)
print(importance_df)

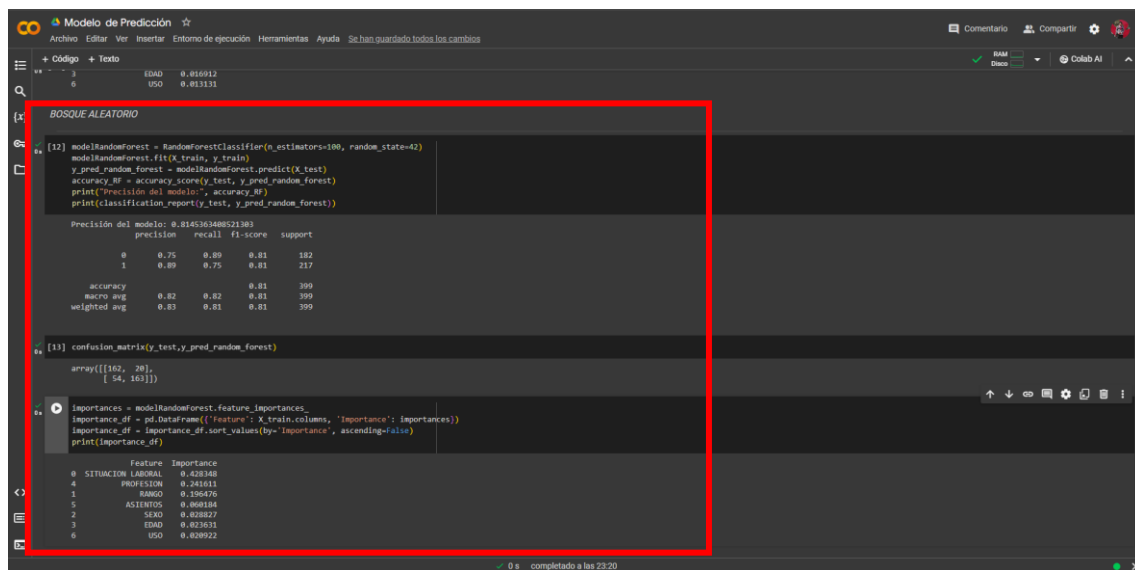
RESULTADO

MOSTRAR RESULTADOS

[ ] from sklearn.metrics import roc_curve, roc_auc_score, auc
import matplotlib.pyplot as plt

[ ] y_pred_logistic = modelLogisticRegression.predict_proba(X_test)[:, 1]
```

Ejecución Exitosa



```
Importance_df = importance_df.sort_values(by='Importance', ascending=False)
print(importance_df)

Feature Importance
0 SITUACION LABORAL 0.468064
4 PROFESION 0.228787
1 RANGO 0.209895
5 ASIENTOS 0.066931
2 SEXO 0.025958
3 EDAD 0.016912
6 USO 0.013131

BOSQUE ALEATORIO

[12] modelRandomForest = RandomForestClassifier(n_estimators=100, random_state=42)
modelRandomForest.fit(X_train, y_train)
y_pred_random_forest = modelRandomForest.predict(X_test)
accuracy_RF = accuracy_score(y_test, y_pred_random_forest)
print("Precisión del modelo:", accuracy_RF)
print(classification_report(y_test, y_pred_random_forest))

Precisión del modelo: 0.8145363408521383
precision    recall  f1-score   support

0     0.75     0.89     0.81     182
1     0.89     0.75     0.81     217

accuracy          0.81     399
macro avg         0.82     0.82     0.81     399
weighted avg      0.83     0.81     0.81     399

[13] confusion_matrix(y_test, y_pred_random_forest)

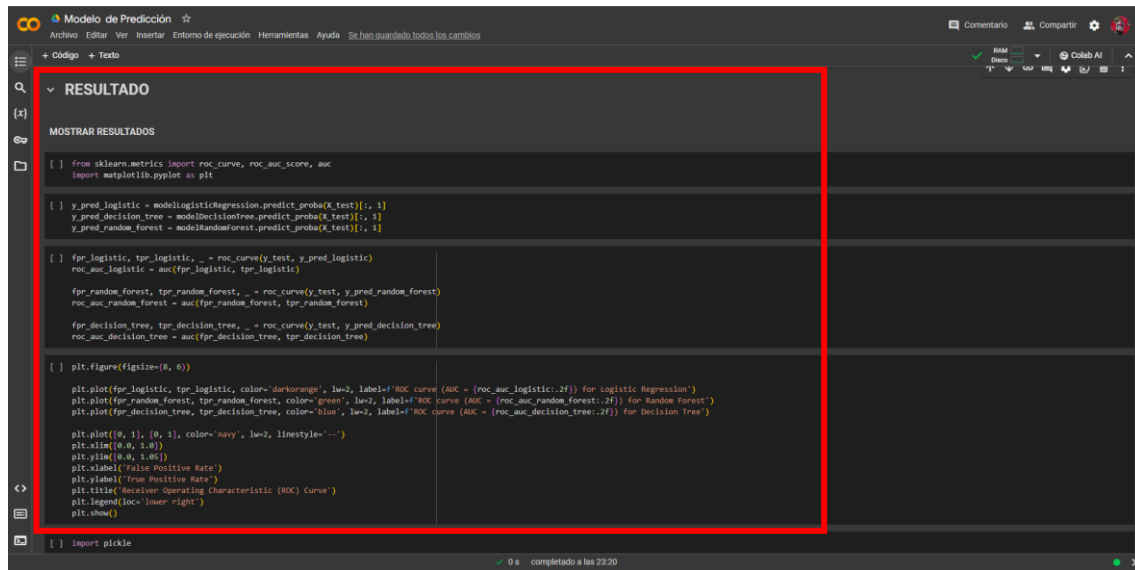
array([[162, 20],
       [ 54, 163]])

Importances = modelRandomForest.feature_importances_
importance_df = pd.DataFrame({'feature': X_train.columns, 'importance': importances})
importance_df = importance_df.sort_values(by='Importance', ascending=False)
print(importance_df)

Feature Importance
0 SITUACION LABORAL 0.428348
4 PROFESION 0.241611
1 RANGO 0.196476
5 ASIENTOS 0.068184
2 SEXO 0.038527
3 EDAD 0.023631
6 USO 0.020922
```

Paso 9: Ejecución de resultados (CP009)

Ejecutar bloque de “Resultados”



```
[ ] from sklearn.metrics import roc_curve, roc_auc_score, auc
import matplotlib.pyplot as plt

[ ] y_pred_logistic = modelLogisticRegression.predict_proba(X_test)[1, :]
y_pred_decision_tree = modelDecisionTree.predict_proba(X_test)[1, :]
y_pred_random_forest = modelRandomForest.predict_proba(X_test)[1, :]

[ ] fpr_logistic, tpr_logistic, _ = roc_curve(y_test, y_pred_logistic)
roc_auc_logistic = auc(fpr_logistic, tpr_logistic)

fpr_random_forest, tpr_random_forest, _ = roc_curve(y_test, y_pred_random_forest)
roc_auc_random_forest = auc(fpr_random_forest, tpr_random_forest)

fpr_decision_tree, tpr_decision_tree, _ = roc_curve(y_test, y_pred_decision_tree)
roc_auc_decision_tree = auc(fpr_decision_tree, tpr_decision_tree)

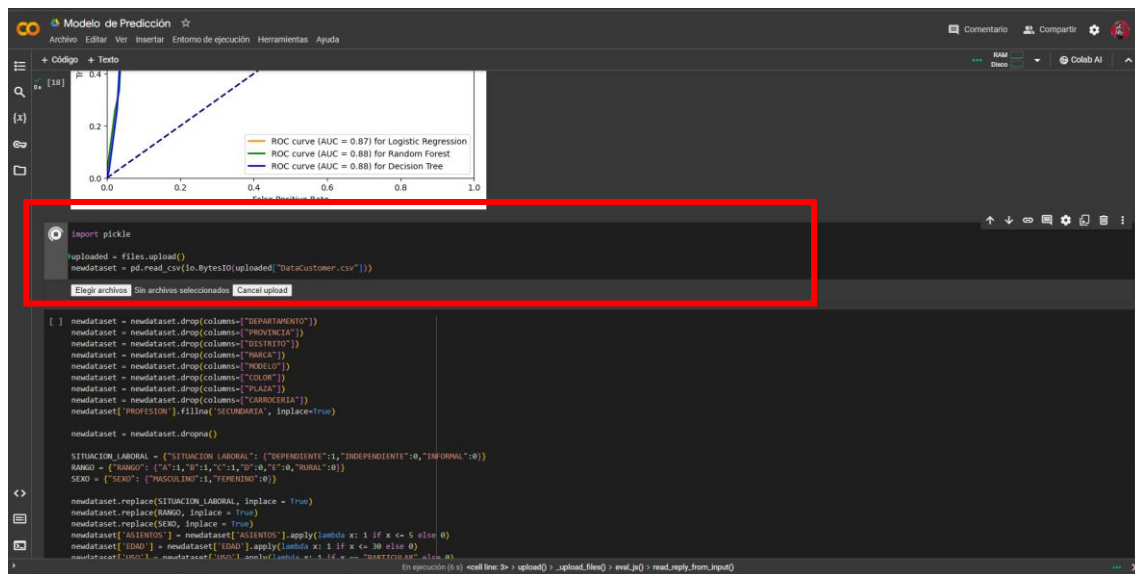
[ ] plt.figure(figsize=(8, 6))

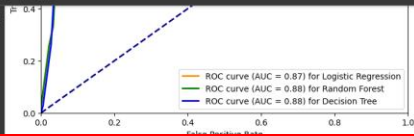
plt.plot(fpr_logistic, tpr_logistic, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc_logistic:.2f}) for Logistic Regression')
plt.plot(fpr_random_forest, tpr_random_forest, color='green', lw=2, label=f'ROC curve (AUC = {roc_auc_random_forest:.2f}) for Random Forest')
plt.plot(fpr_decision_tree, tpr_decision_tree, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc_decision_tree:.2f}) for Decision Tree')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()

[ ] import pickle
```

Se le pedirá seleccionar un archivo



```
[18] 

[ ] import pickle

uploaded = files.upload()
newdataset = pd.read_csv(io.BytesIO(uploaded["DataCustomer.csv"]))

Elegir archivos Sin archivos seleccionados Cancel upload

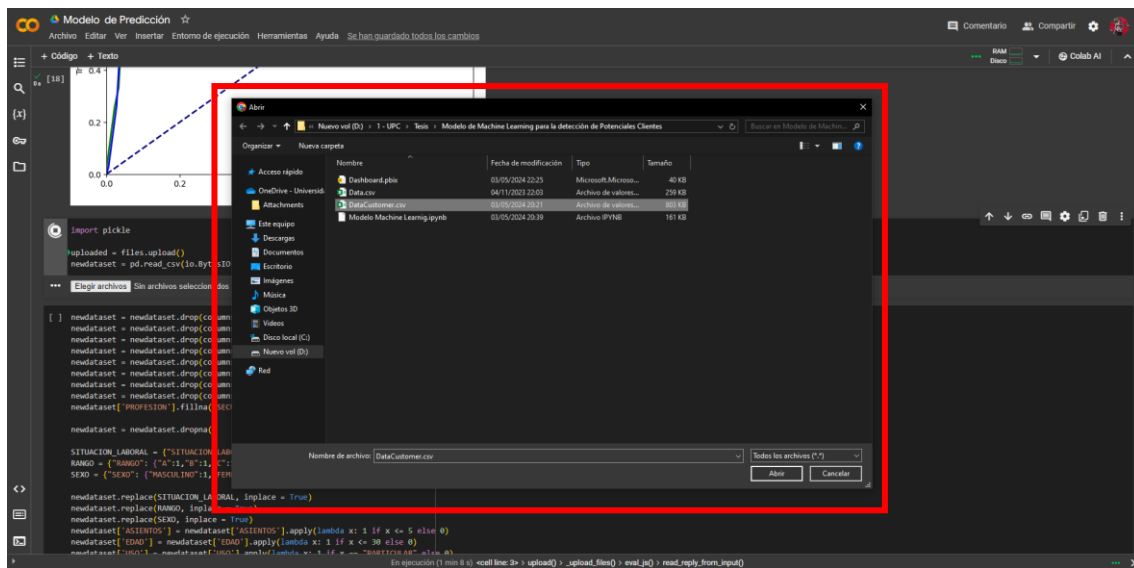
[ ] newdataset = newdataset.drop(columns=["DEPARTAMENTO"])
newdataset = newdataset.drop(columns=["PROVINCIA"])
newdataset = newdataset.drop(columns=["DISTRITO"])
newdataset = newdataset.drop(columns=["MARCA"])
newdataset = newdataset.drop(columns=["MODELO"])
newdataset = newdataset.drop(columns=["COLOR"])
newdataset = newdataset.drop(columns=["PLAZA"])
newdataset = newdataset.drop(columns=["CARROceria"])
newdataset["PROFESION"].fillna("SECCIONARIA", inplace=True)

newdataset = newdataset.dropna()

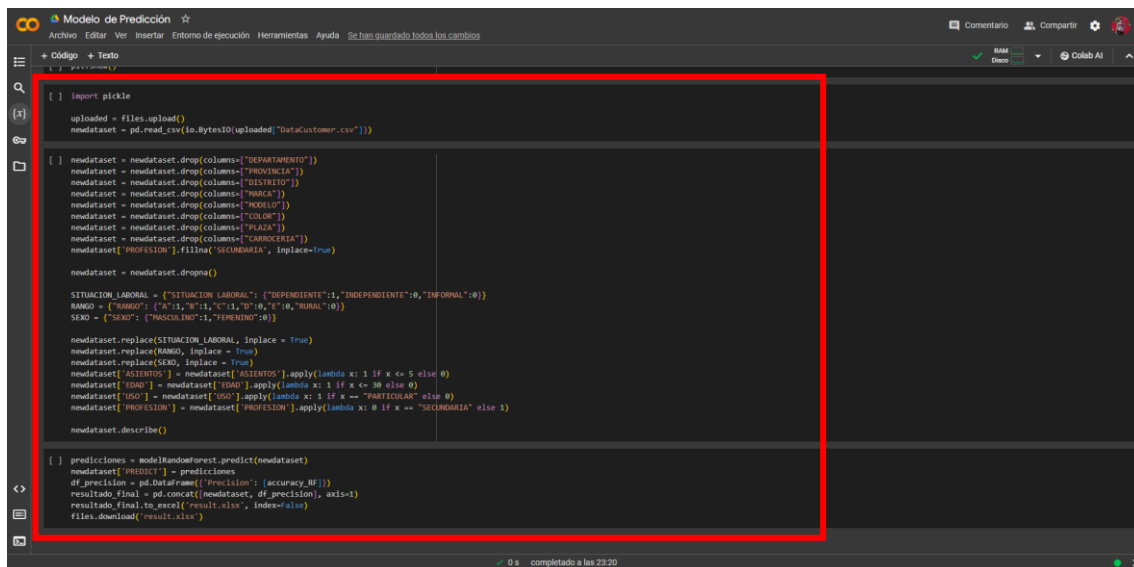
SITUACION_LABORAL = {"SITUACION_LABORAL": {"DEPENDIENTE":1, "INDEPENDIENTE":0, "INFORMAL":0}}
RANGO = {"RANGO": {"A":1, "B":1, "C":1, "D":0, "E":0, "RURAL":0}}
SEXO = {"SEXO": {"MASCULINO":1, "FEMENINO":0}}

newdataset.replace(SITUACION_LABORAL, inplace = True)
newdataset.replace(RANGO, inplace = True)
newdataset.replace(SEXO, inplace = True)
newdataset["ASIENTOS"] = newdataset["ASIENTOS"].apply(lambda x: 1 if x <= 5 else 0)
newdataset["EDAD"] = newdataset["EDAD"].apply(lambda x: 1 if x <= 30 else 0)
newdataset["PROFESION"] = newdataset["PROFESION"].apply(lambda x: 1 if x == "SECCIONARIA" else 0)
```

Subir archivo “DataCustomer.csv”



Esperar que cargue el archivo y continuar ejecutando el bloque de código



Se le descarga el archivo result.xlsx, guardarlo porque será utilizado después.

Modelo de Predicción

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

+ Código + Texto

```
[26] newdataset.replace(SEXO, inplace = True)
newdataset['ASIENTOS'] = newdataset['ASIENTOS'].apply(lambda x: 1 if x <= 5 else 0)
newdataset['EDAD'] = newdataset['EDAD'].apply(lambda x: 1 if x <= 30 else 0)
newdataset['USO'] = newdataset['USO'].apply(lambda x: 1 if x == "HISTORIAS" else 0)
newdataset['PROFESION'] = newdataset['PROFESION'].apply(lambda x: 0 if x == "SECUNDARIA" else 1)

newdataset.describe()
```

	SITUACION LABORAL	RANGO	SEXO	EDAD	PROFESION	ASIENTOS	USO
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	0.733184	0.843797	0.600897	0.809417	0.789238	0.915546	0.913303
std	0.442461	0.363184	0.489897	0.392908	0.408002	0.278172	0.281495
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000
50%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
predicciones = modelRandomforest.predict(newdataset)
newdataset['PREDICT'] = predicciones
df_precision = pd.DataFrame({'precision': [accuracy_RF]})
resultado_final = pd.concat([newdataset, df_precision], axis=1)
resultado_final.to_excel('result.xlsx', index=False)
files.download('result.xlsx')
```

0 s - completado a las 23:25

result (1).xlsx
25.5 KB • Últ.

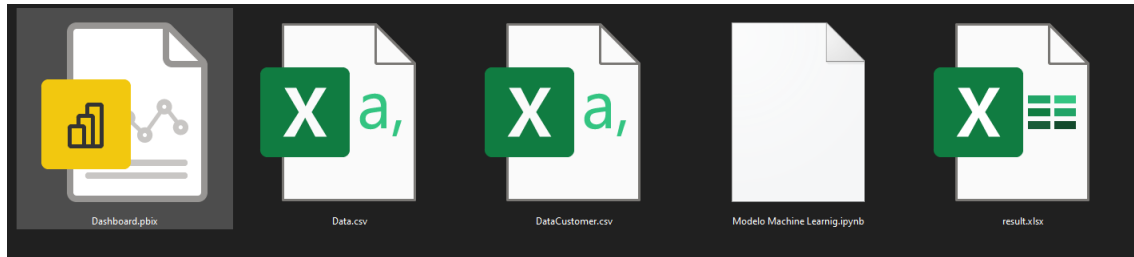
Modelo_de_Predicción (5).ipynb
17.3 KB • Hace 20 minutos

Modelo de Machine Learning para la
detección de Potenciales Clientes.rar
213 KB • Hace 51 minutos

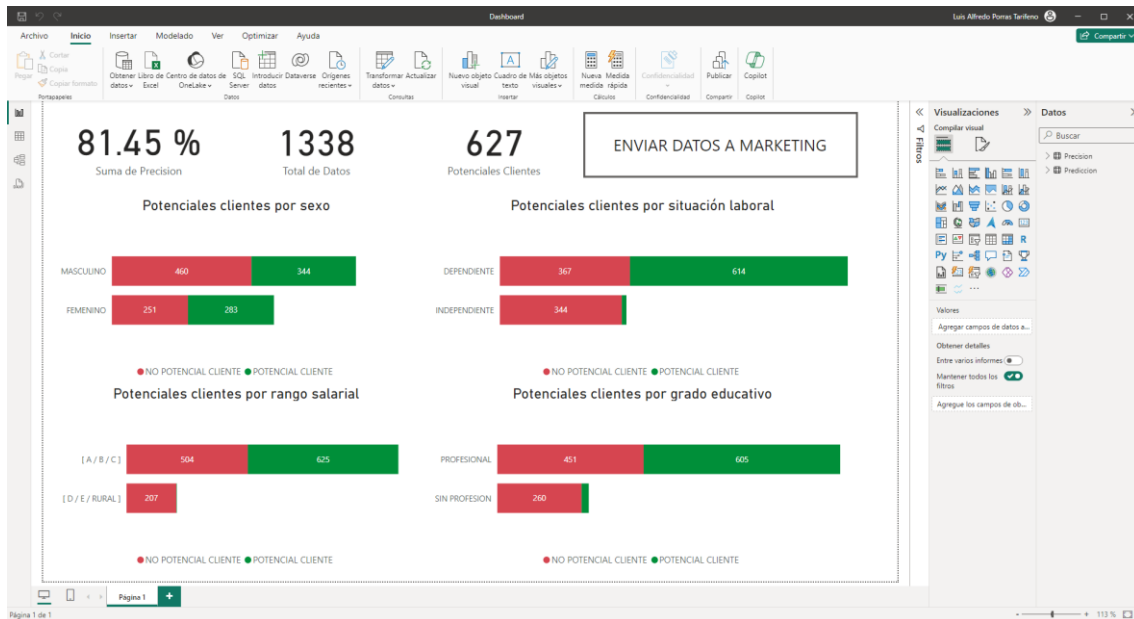
Modelo_de_Predicción (4).ipynb
140 KB • Hace 1 hora

Paso 10: Cargar los resultados al Power BI (CP010)

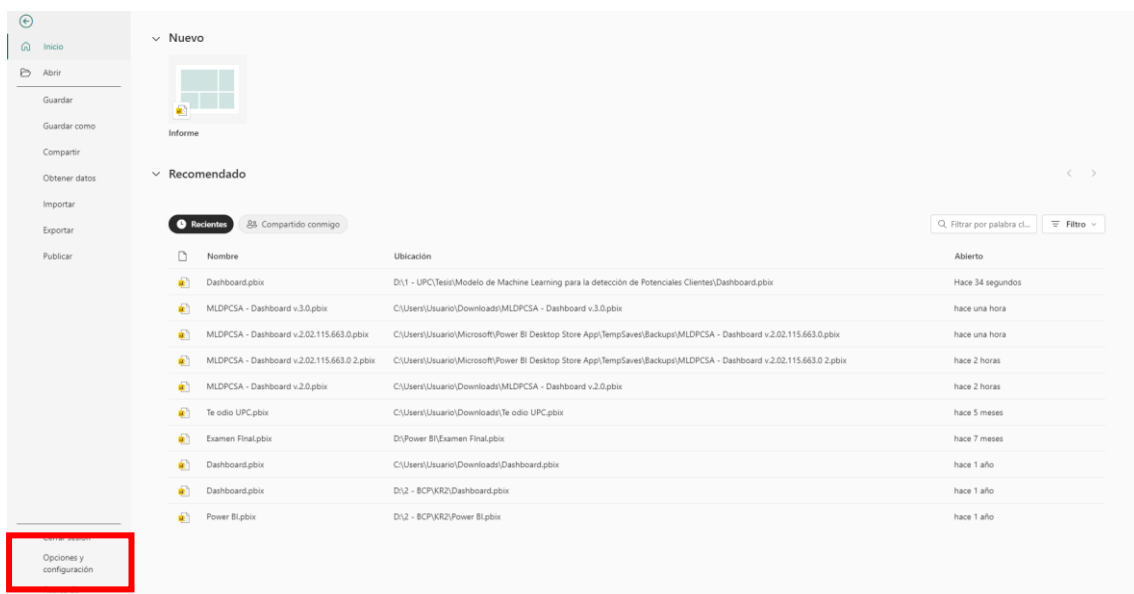
Abrir el archivo “Dashboard.pbix”



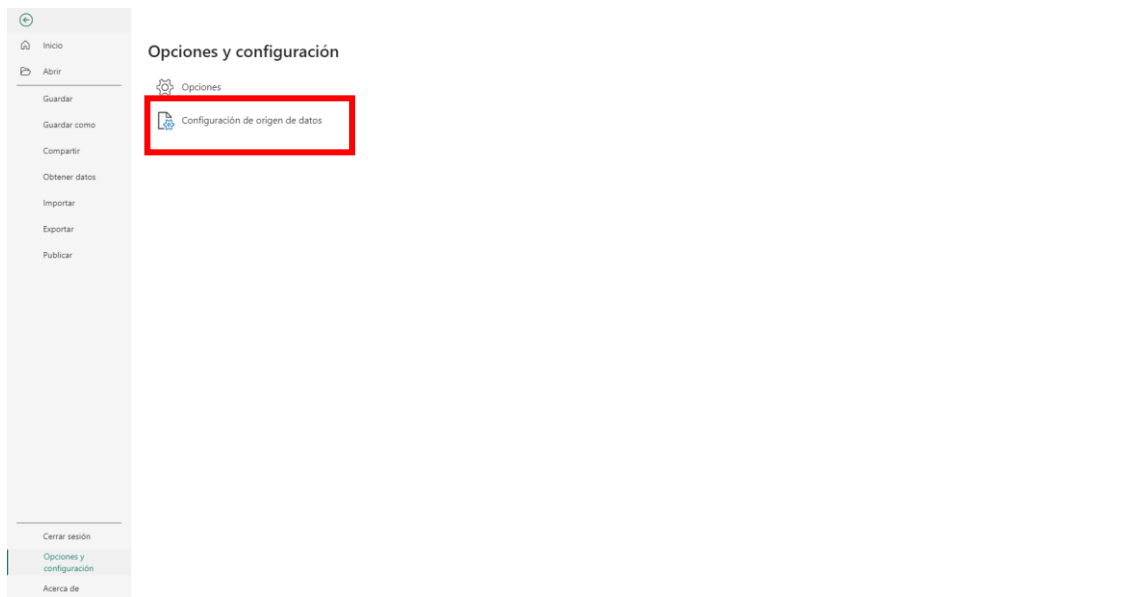
Se abrirá el dashboard en Power BI



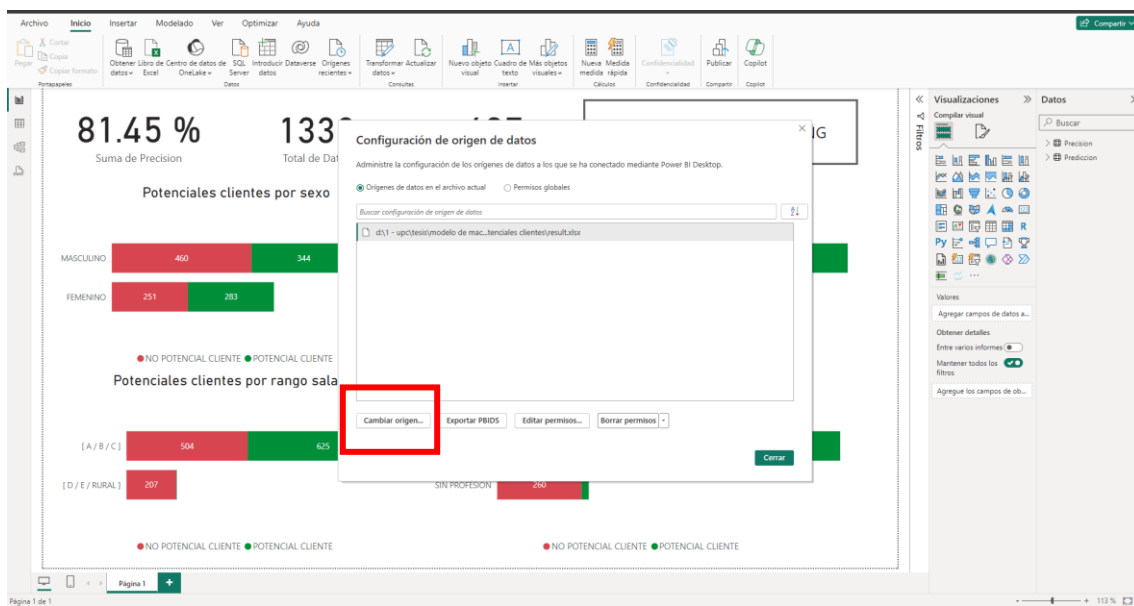
Se debe seleccionar el nuevo origen de datos. Dar click en “Archivo” > “Opciones y Configuraciones”



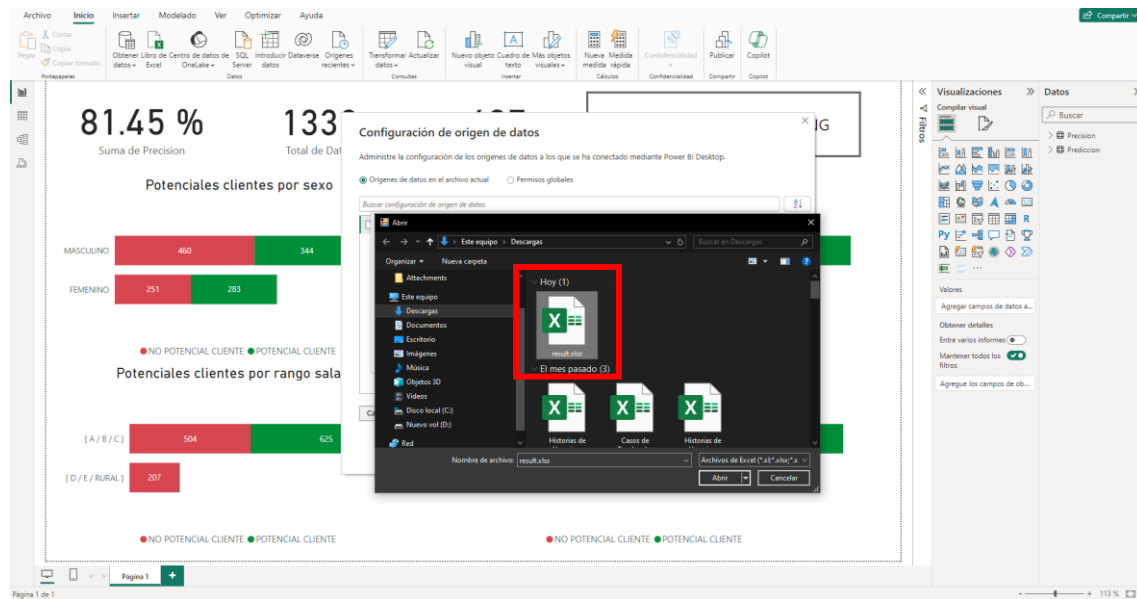
Dar click en “Configuración de Origen de datos”



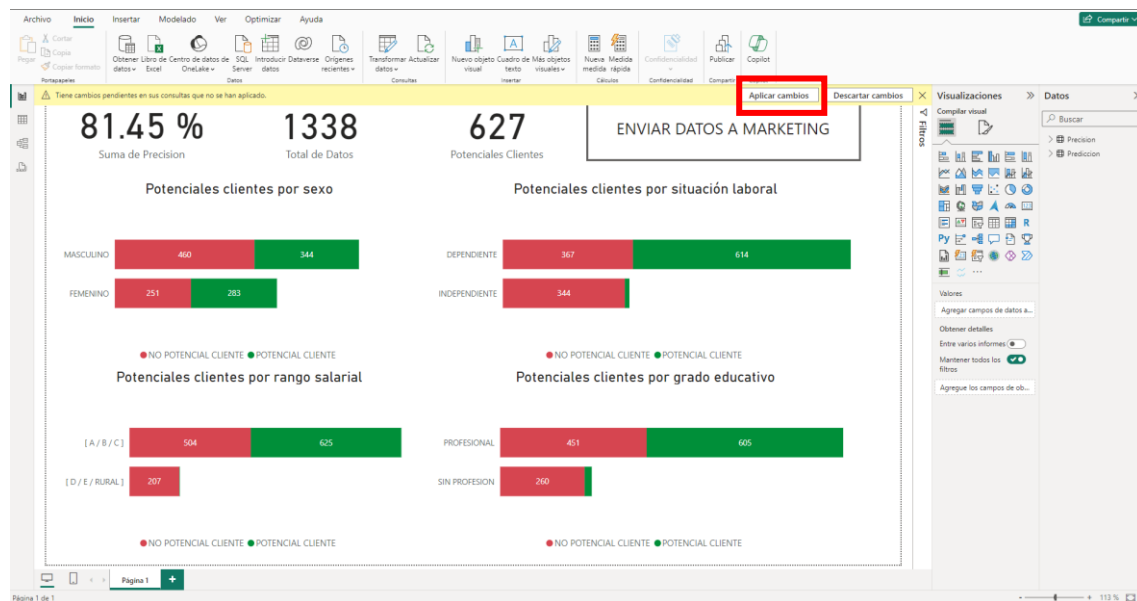
Dar click en Cambiar origen



Seleccionar el archivo “result.xlsx” obtenido del modelo

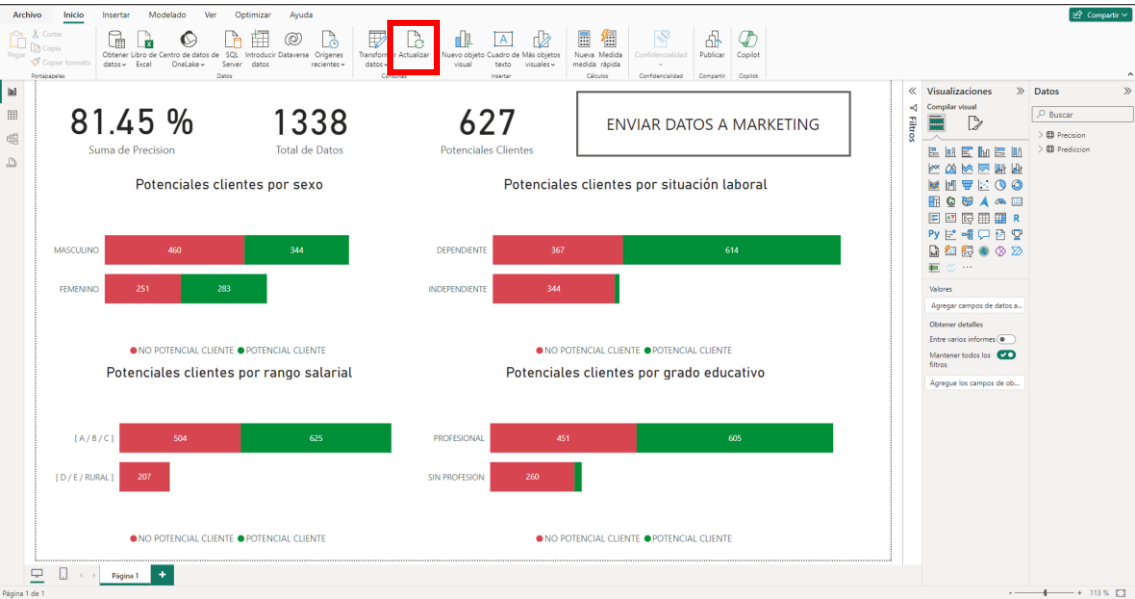


Dar click en “Aplicar Cambios”



Paso 11: Actualizar los resultados al Power BI (CP011)

Dar click en “actualizar”



Carga Exitosa

