



UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

FACULTAD DE INGENIERÍA

PROGRAMA ACADÉMICO DE INGENIERÍA DE SISTEMAS DE INFORMACIÓN

MODELO DE MACHINE LEARNING PARA IDENTIFICAR POTENCIALES
CLIENTES DE SEGUROS DE AUTOMÓVILES

Manual de Despliegue

AUTORES

Porras Tarifeño, Luis Alfredo

Medina Cortez, Alexander Fausto

ASESORES

Rodriguez Castillo, Hugo Maximiliano

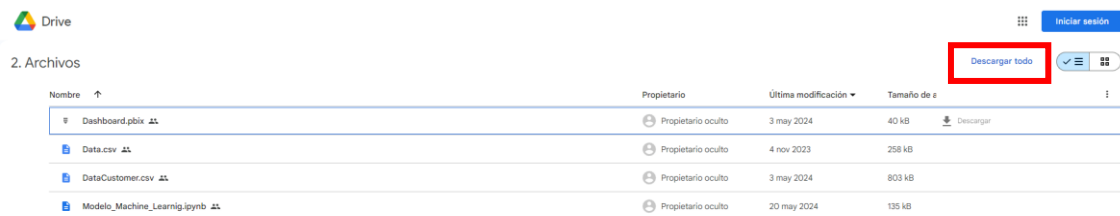
Lima, 2024

Requerimientos mínimos para la ejecución del modelo:

En este apartado detallaremos los requerimientos para el despliegue del algoritmo (script) y Dashboard.

Archivos:

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



Cuenta Google:

- Como mínimo debe de contar con una cuenta Google para acceder al Google Colab
- **(Opcional)** Si desea publicar el Dashboard en la web, necesita tener un plan PRO o Premium, sino lo podrá ver en la vista on premise descargando el Power Bi Desktop.

Cuenta gratuita	Power BI Pro	Power BI Premium por usuario	Power BI en Microsoft Fabric
Cree informes completos e interactivos con análisis visuales a su alcance de forma gratuita.	Acceda a informes de Power BI compartidos con usted y publique los suyos para un impacto aún mayor.	Conceda licencias a usuarios específicos con características de escala empresarial.	Comparta informes con usuarios que no tengan licencias de pago y obtenga acceso a las cargas de trabajo de Microsoft Fabric con una sola capacidad.
Gratuito	9,40 € por usuario al mes <small>El precio no incluye IVA.</small>	18,70 € por usuario al mes <small>El precio no incluye IVA.</small>	Variable
Empezar gratis	Comprar ahora	Comprar ahora	Consultar los precios

Power BI:

Descargar el Power BI desktop:


- Paso 1: Ingrese a la ruta: <https://www.microsoft.com/es-es/download/details.aspx?id=58494>
- Paso 2: Click en botón descargar:

Maximiza el día a día con Microsoft 365

Obtén protección online, almacenamiento seguro en la nube y aplicaciones innovadoras diseñadas para adaptarse a tus necesidades, todo en un solo plan.

Para 1 usuario

Para un máximo de 6 usuarios



Microsoft Power BI Desktop

Microsoft Power BI Desktop se creó para los analistas. Combina visualizaciones interactivas de última generación con consultas y modelado de datos líderes en la industria integrados. Cree y publique sus informes en Power BI. Power BI Desktop ayuda a facilitar a otras personas información fundamental puntual, en cualquier momento y desde cualquier lugar.

Important! Selecting a language below will dynamically change the complete page content to that language.

Seleccionar idioma

Español

Descargar

Essential BI | Follow us all

- Paso 3: Seleccionar el tipo “PBIDesktopSetupx64.exe” y click en descargar:

Elige la descarga que deseas

☐ Nombre del archivo

Tamaño

☐ PBIDesktopSetup.exe

456.0 MB

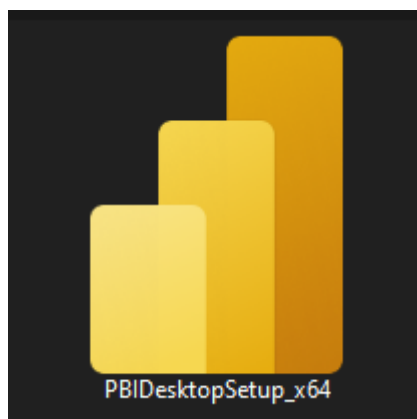
☒ PBIDesktopSetup_x64.exe

498.8 MB

Descargar

Tamaño total: 498.8 MB

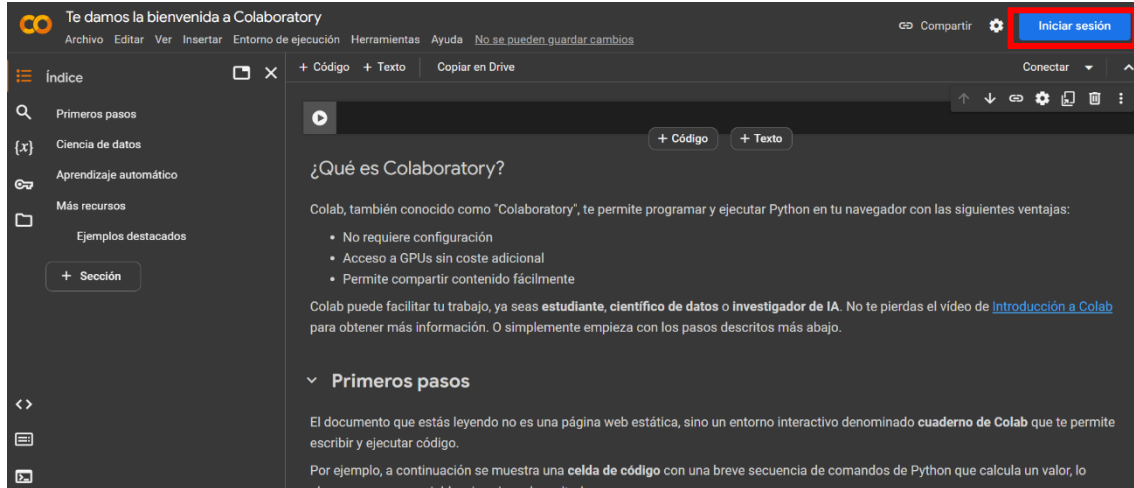
- Paso 4: Esperar la descargar y ejecutar el programa:



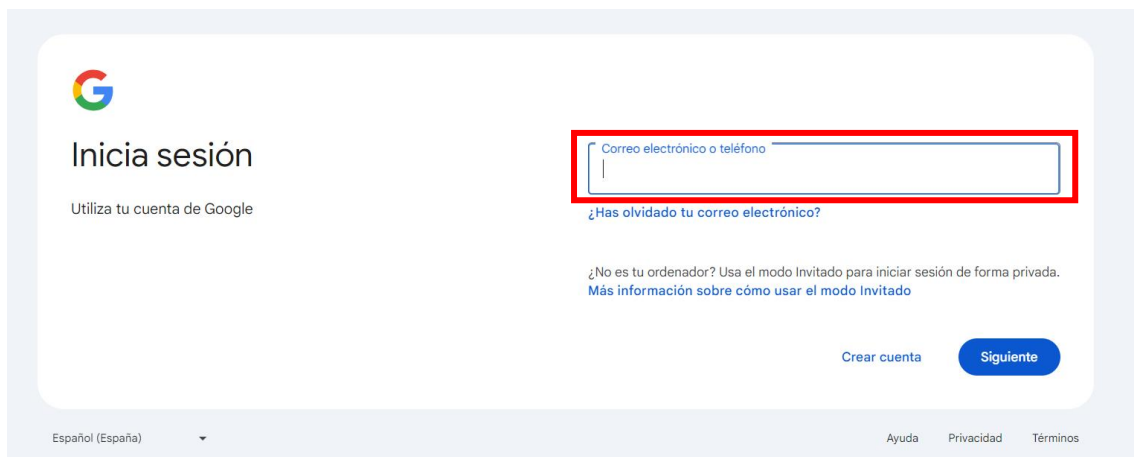
Algoritmo de Machine Learning (Script)

Paso 1: Login en Google Colab

1.1. Dar Click en iniciar sesión en Google Colab: <https://colab.research.google.com/>



1.2. Ingresar tu correo electrónico



1.3. Ingresar tu contraseña



1.4. Sesión Exitosa

Te damos la bienvenida a Colaboratory

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

os Compartir

Índice

Primeros pasos

Ciencia de datos

Aprendizaje automático

Más recursos

Ejemplos destacados

+ Sección

+ Código + Texto Copiar en Drive

Conectar Colab AI

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.

3 Cool Google Colab Features

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

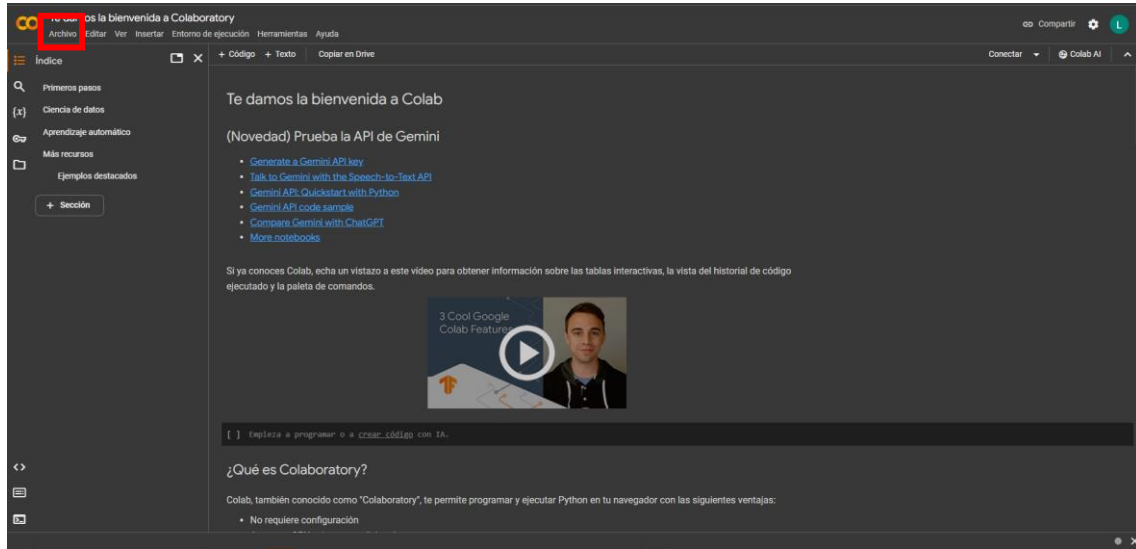
¿Qué es Colaboratory?

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

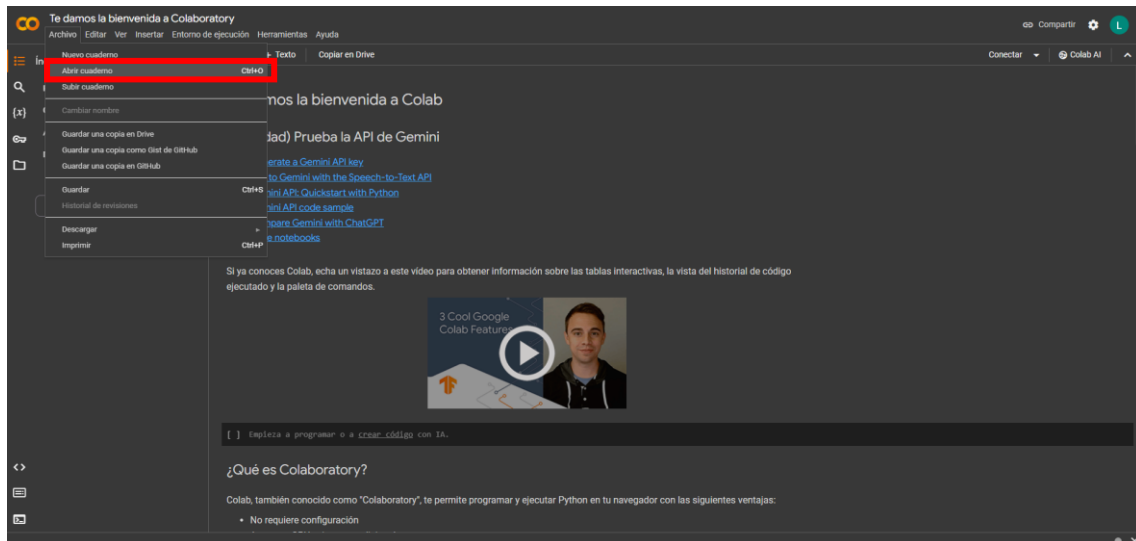
- No requiere configuración

Paso 2: Cargar el algoritmo a Google Colab

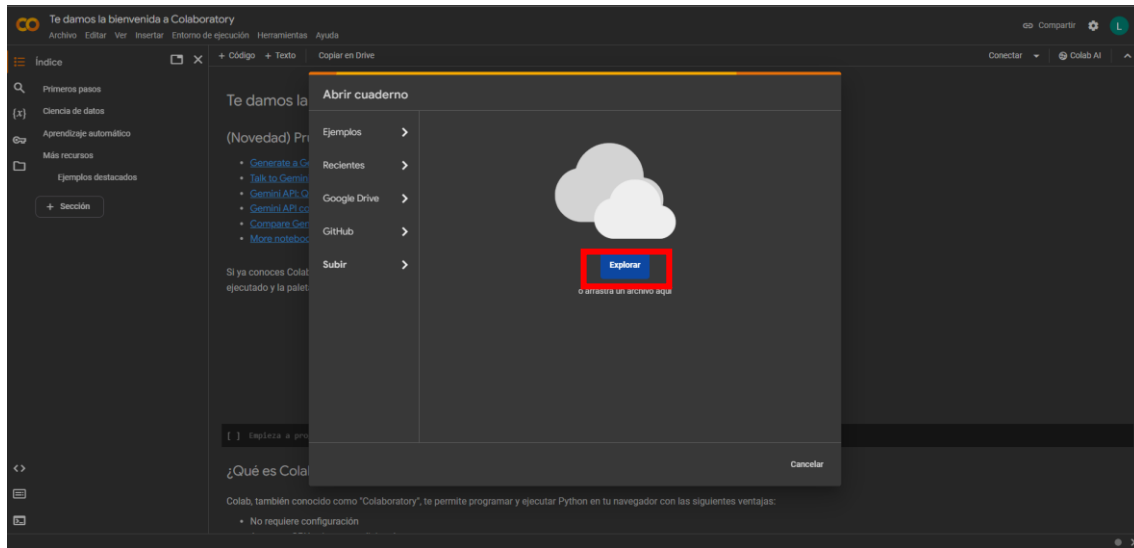
2.1. Dar click en “Archivo”



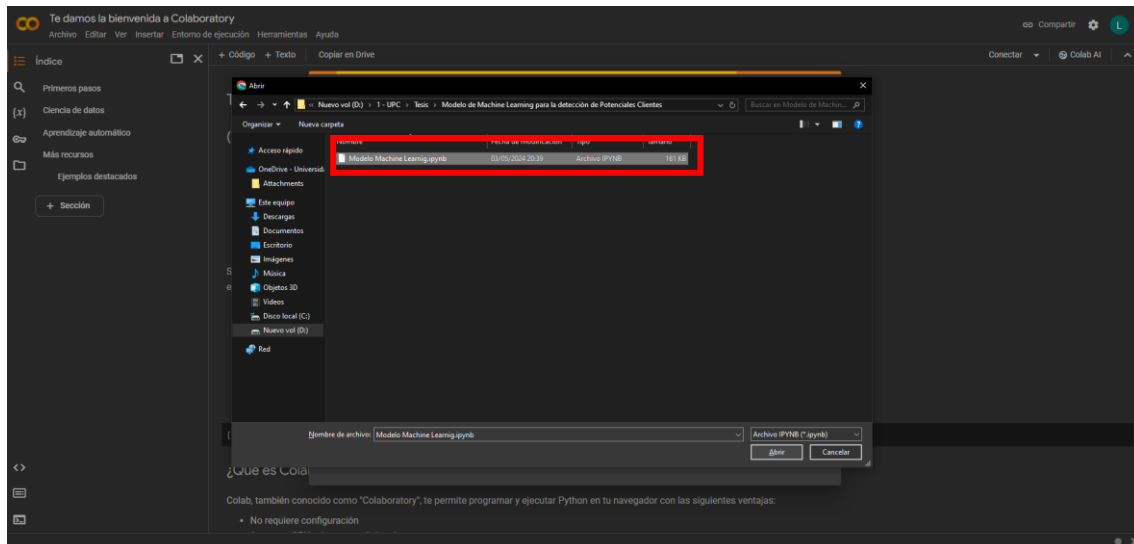
2.2. Dar click en “Abrir cuaderno”



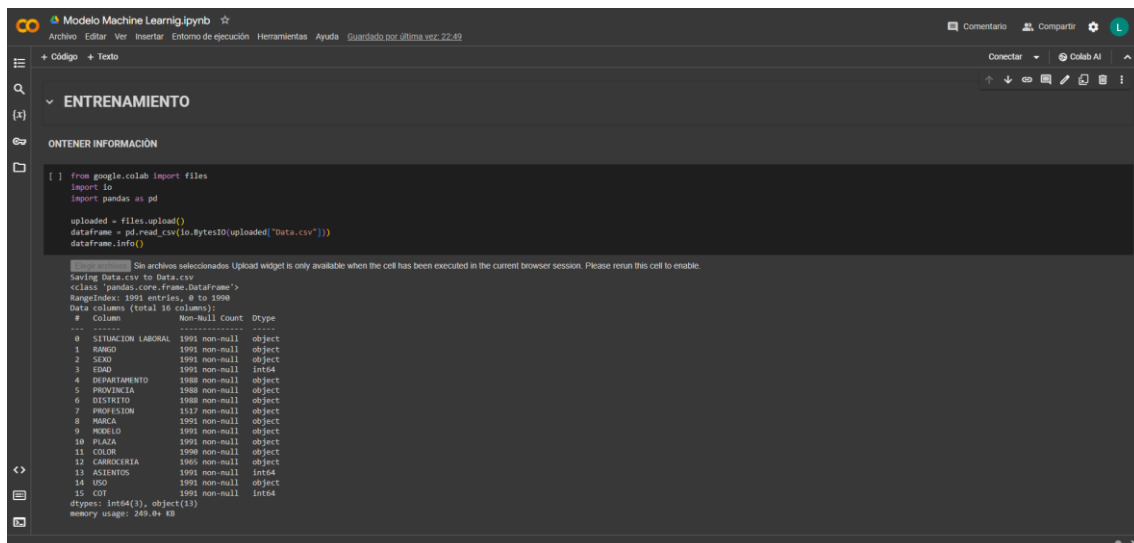
2.3. Dar click en “Subir” > “Explorar”



2.4. Subes el archivo “Modelo Machine Learning.ipynb”

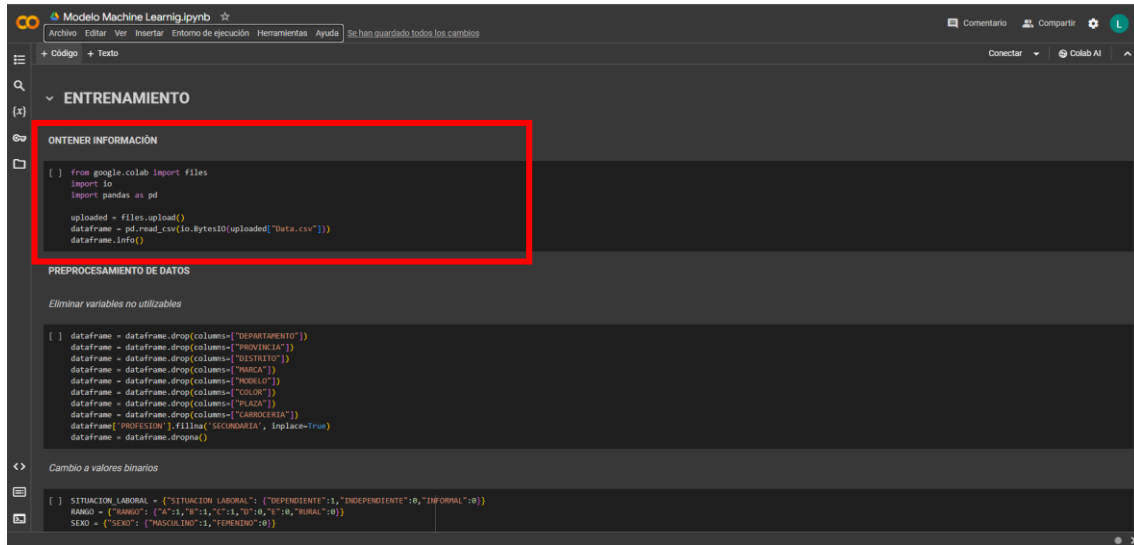


2.5. Carga exitosa



Paso 3: Cargar el algoritmo dataset

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



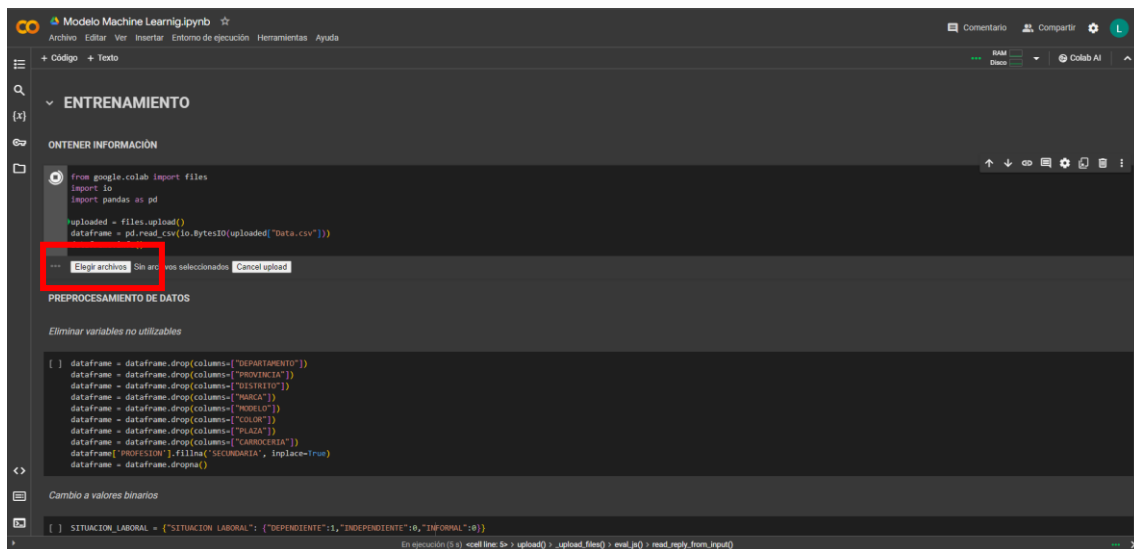
The screenshot shows the Google Colab interface for a notebook titled 'Modelo Machine Learning.ipynb'. The 'ENTRENAMIENTO' section is expanded, and the 'OBTENER INFORMACIÓN' block is highlighted with a red box. The code in this block is as follows:

```
[ ] 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()
```

Below this block, the 'PREPROCESAMIENTO DE DATOS' section is visible, containing code for dropping columns and handling missing values.

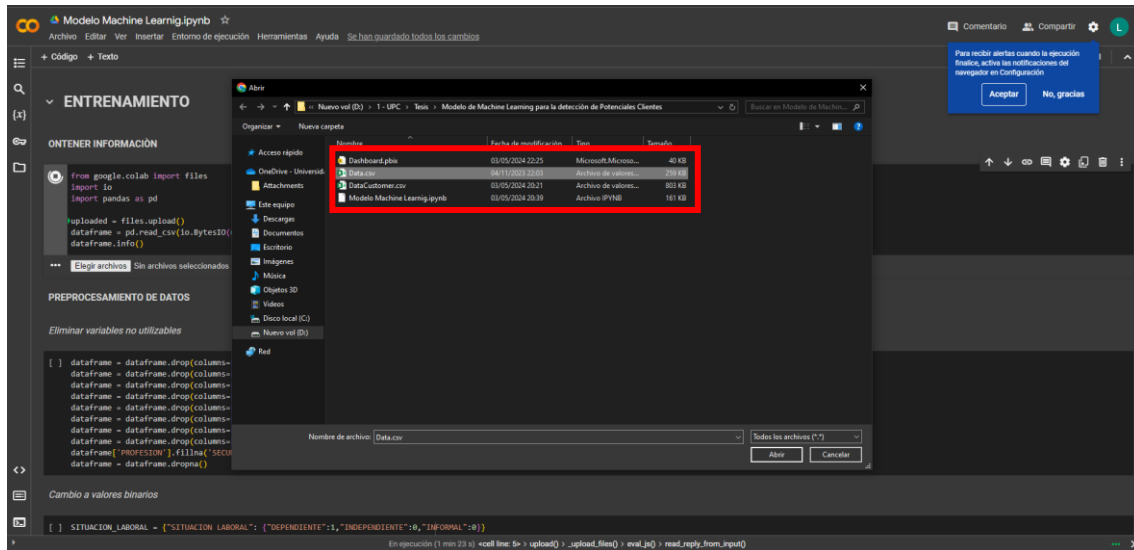
3.2. Dar click en “Elegir Archivo”



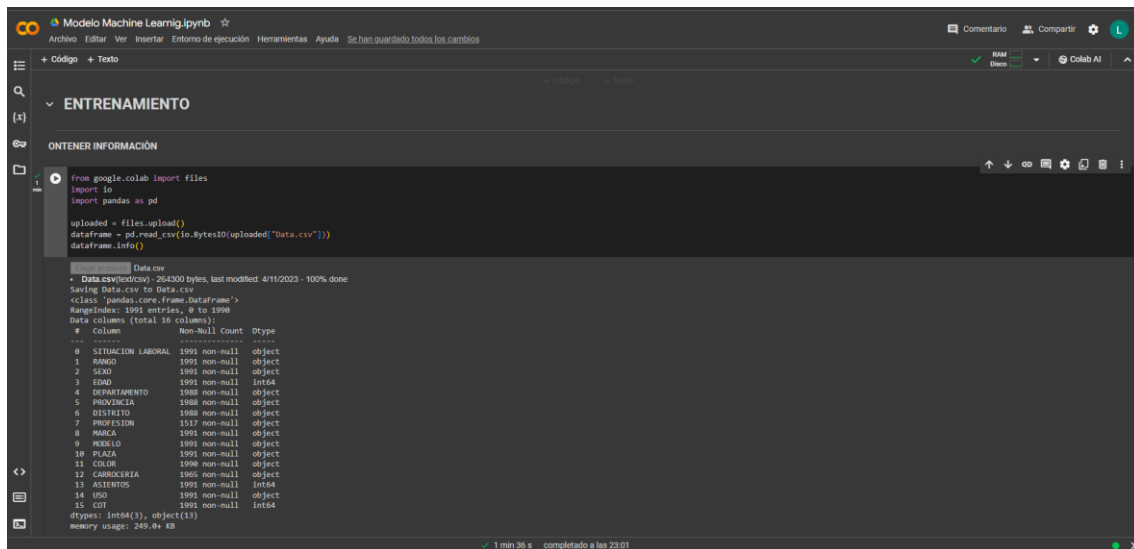
The screenshot shows the Google Colab interface for the same notebook. The 'OBTENER INFORMACIÓN' block is now in a state where a file upload dialog is visible. The 'Elegir archivos' button is highlighted with a red box. The code in the block is the same as in the previous screenshot.

Below the code, the 'PREPROCESAMIENTO DE DATOS' section is visible, containing code for dropping columns and handling missing values.

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

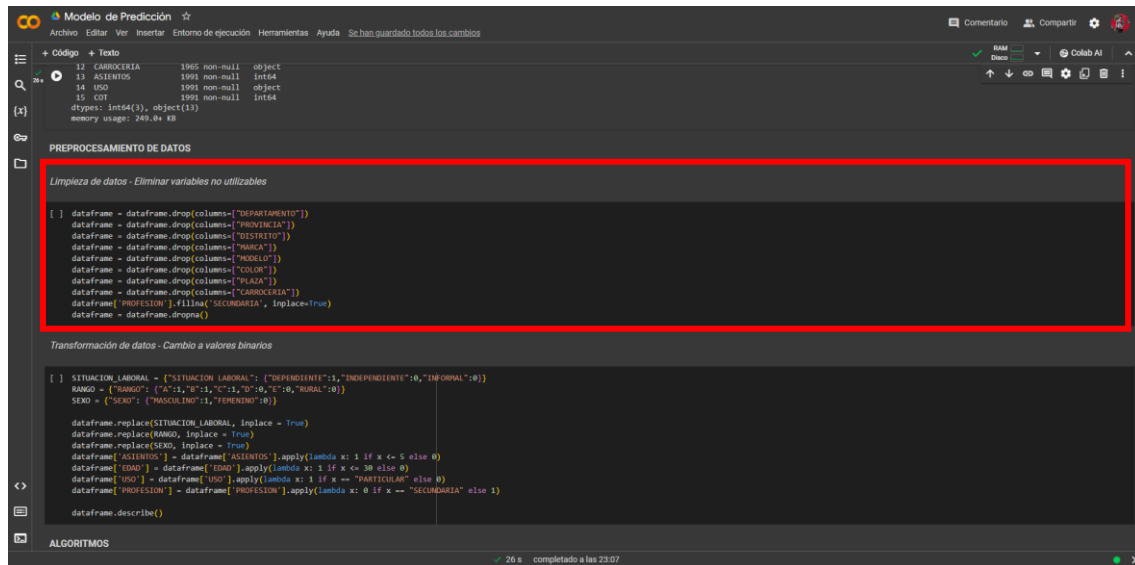


3.4. Carga exitosa



Paso 4: Ejecución del algoritmo de limpieza de datos

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



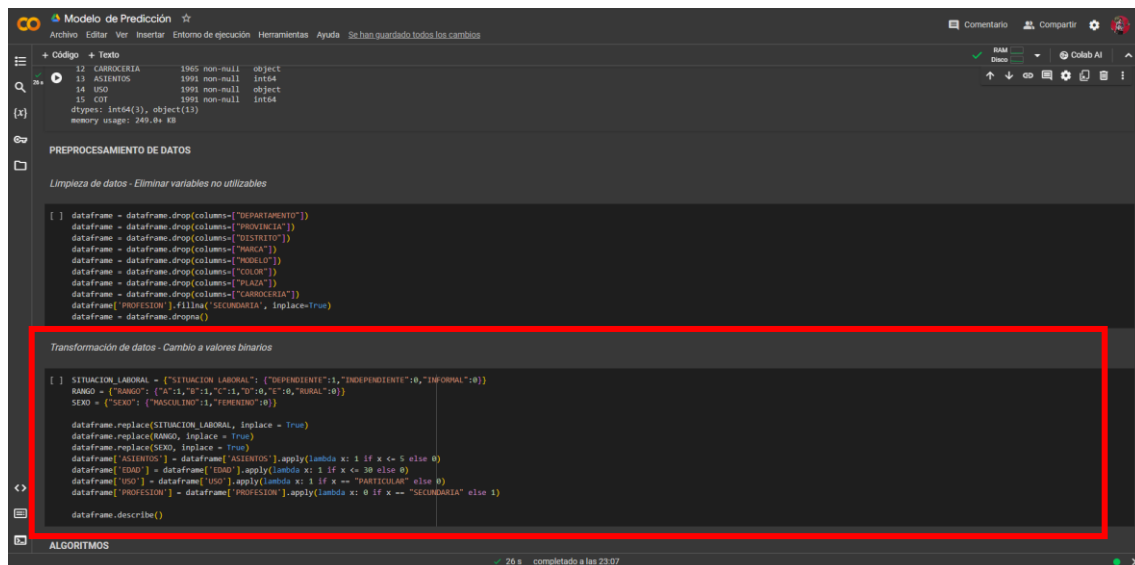
The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes the Google Colab logo and menu options like 'Archivo', 'Editar', 'Ver', 'Insertar', 'Entorno de ejecución', 'Herramientas', and 'Ayuda'. Below the top bar, there's a section for 'PREPROCESAMIENTO DE DATOS'. The first block, 'Limpieza de datos - Eliminar variables no utilizables', is highlighted with a red rectangle. It contains the following Python code:

```
[ ] 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()
```

Below this block is another block titled 'Transformación de datos - Cambio a valores binarios', which contains code for replacing categorical values with binary ones. The bottom of the notebook shows the 'ALGORITMOS' section with a status bar indicating '26 s completado a las 23:07'.

Paso 5: Ejecución del algoritmo de Transformación de datos

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



The screenshot shows the same Jupyter Notebook interface as before, but now the 'Transformación de datos - Cambio a valores binarios' block is highlighted with a red rectangle. This block contains the following Python code:

```
[ ] 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}}

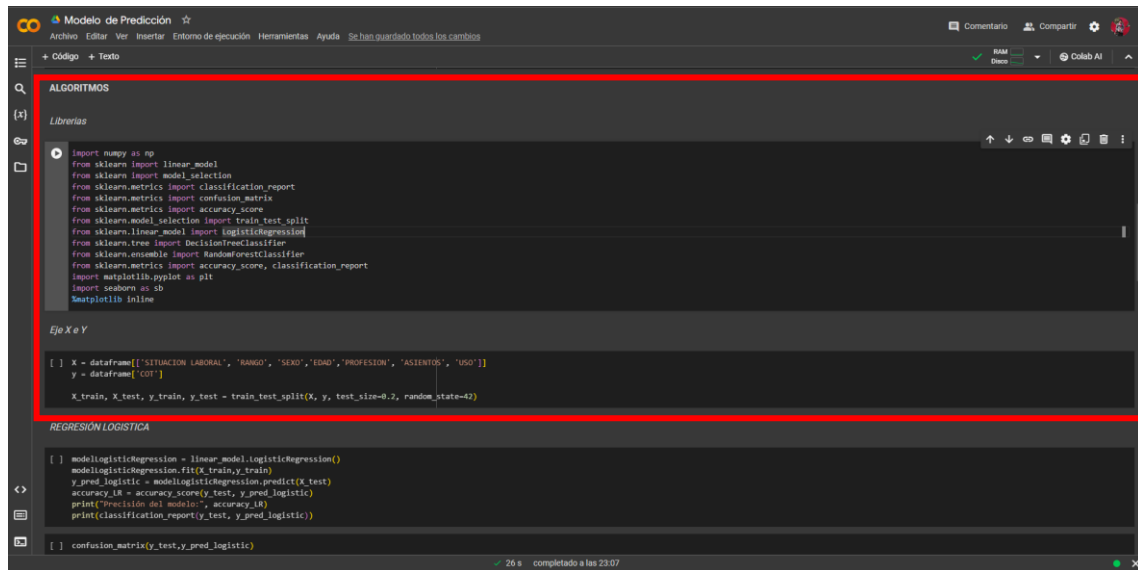
dataframe.replace(SITUACION_LABORAL, inplace=True)
dataframe.replace(RANGO, inplace=True)
dataframe.replace(SEXO, inplace=True)
dataframe["ASIENTOS"] = dataframe["ASIENTOS"].apply(lambda x: 1 if x <= 5 else 0)
dataframe["EDAD"] = dataframe["EDAD"].apply(lambda x: 1 if x <= 30 else 0)
dataframe["USO"]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

The status bar at the bottom indicates '26 s completado a las 23:07'.

Paso 6: Ejecución del algoritmo de Regresión Logística

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



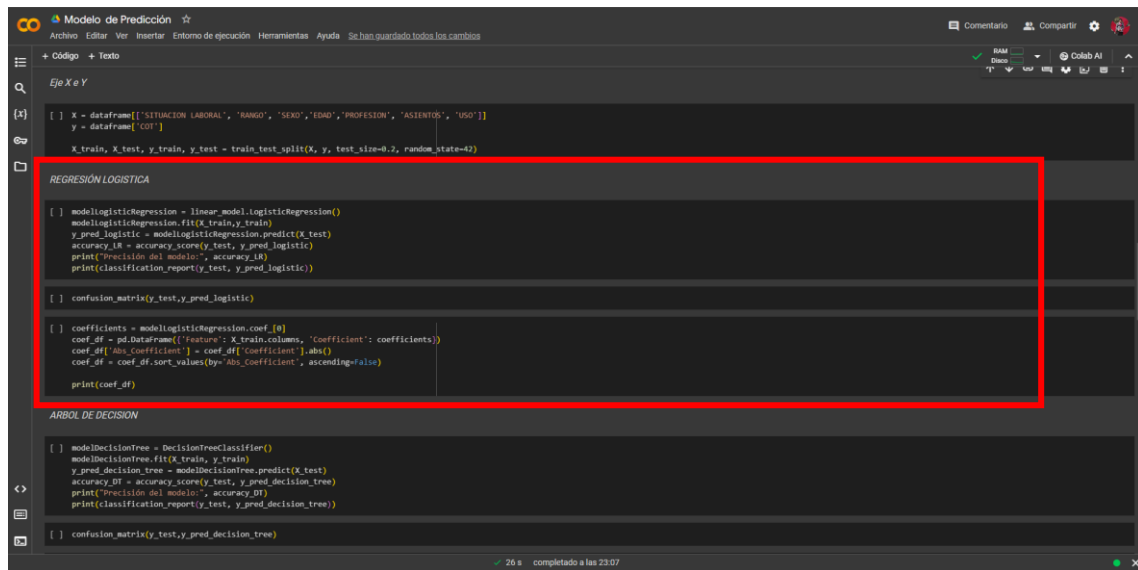
```
import numpy as np
from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model 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)
```

6.2. 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)

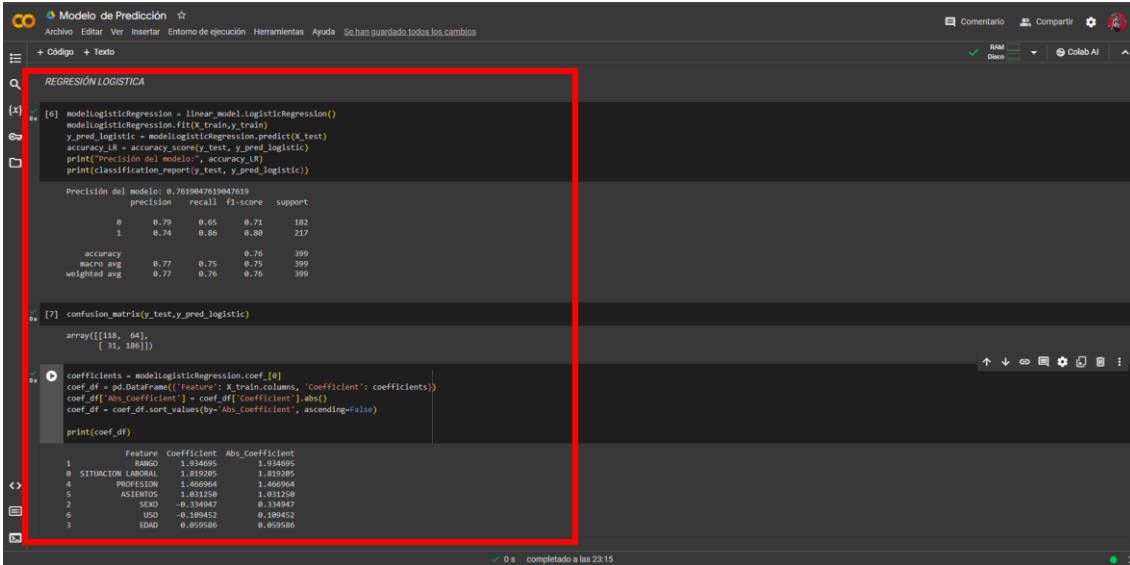
coefficients = modelLogisticRegression.coef_[0]
coef_df = pd.DataFrame({'feature': X_train.columns, 'coefficient': coefficients})
coef_df['Abs_Coefficient'] = coef_df['coefficient'].abs()
coef_df = coef_df.sort_values(by='Abs_Coefficient', ascending=False)
print(coef_df)

# 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)
```

6.3. Ejecución Exitosa



```
Modelo de Predicción
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se han guardado todos los cambios
+ Código + Texto
REGRESIÓN LOGÍSTICA
[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: 0.7019847619847019
              precision    recall  f1-score   support

0               0.79        0.65        0.71        182
1               0.74        0.86        0.80        227

accuracy          0.77        0.75        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]])

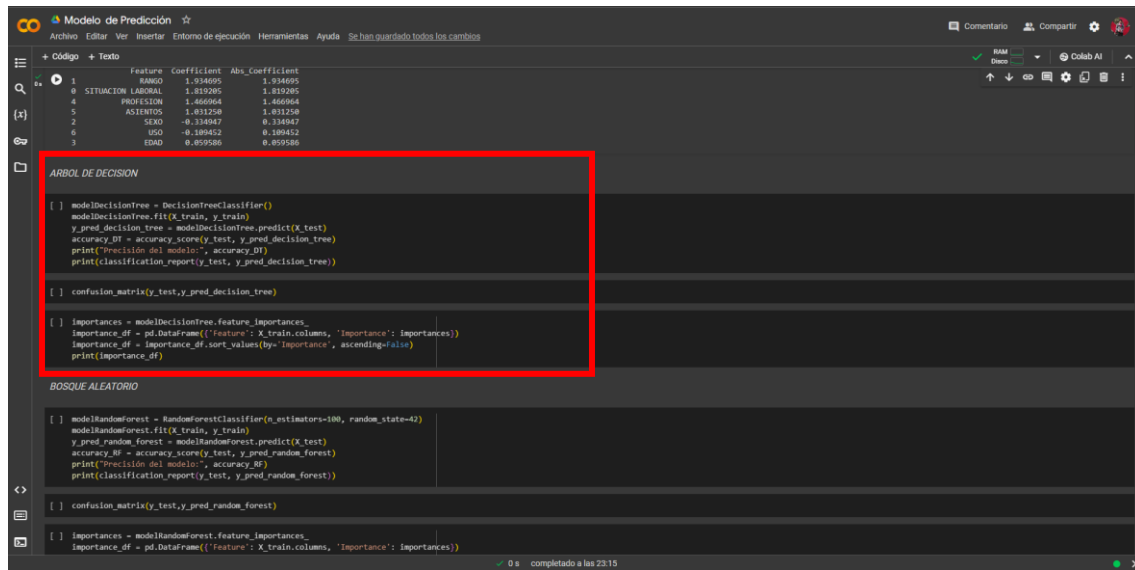
[8] coefficients = modelLogisticRegression.coef_[0]
coef_df = pd.DataFrame({'feature': X_train.columns, 'Coefficient': coefficients})
coef_df['Abs_Coefficient'] = coef_df['Coefficient'].abs()
coef_df = coef_df.sort_values(by='Abs_Coefficient', ascending=False)
print(coef_df)

   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     -0.134647         0.134647
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

7.1. Ejecutar el bloque “Árbol de Decisión”



```
Feature      Coefficient      Abs_Coefficient
0  RANGO      1.934695      1.934695
1  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
```

```
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)

[ ] 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)

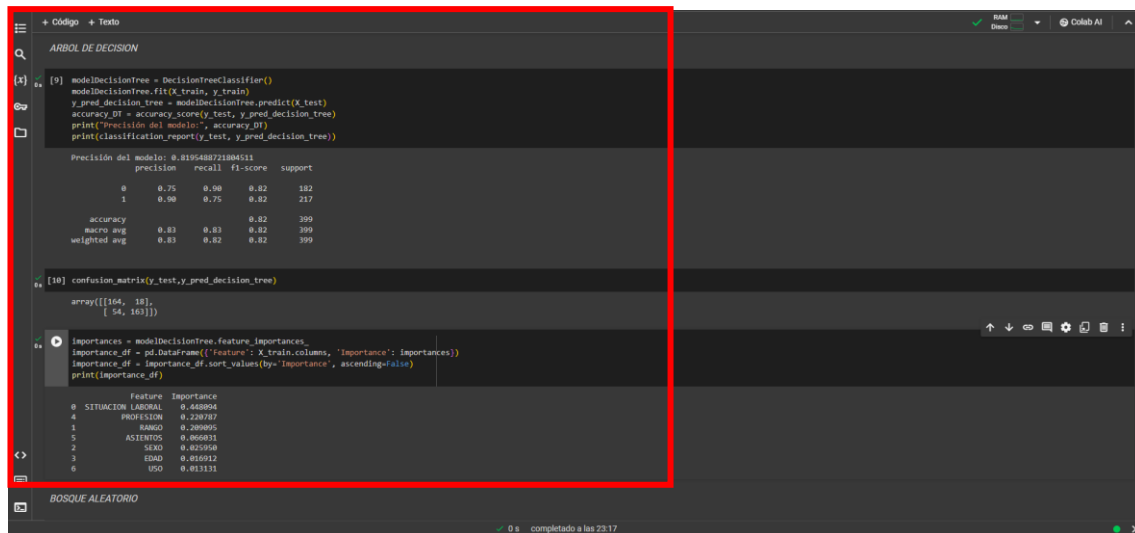
BOSQUE ALEATORIO

[ ] 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})
```

7.2. Ejecución Exitosa



```
ARBOL DE DECISION

[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]])

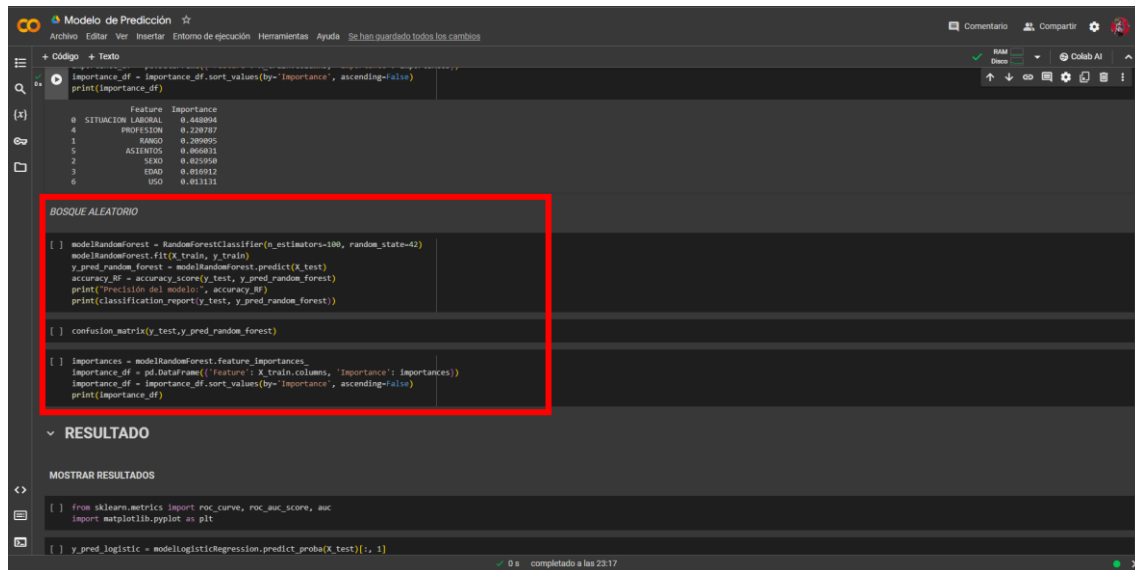
[ ] importances = modelDecisionTree.feature_importances_
importance_df = pd.DataFrame({'feature': X_train.columns, 'importance': importances})
importance_df = importance_df.sort_values(by='importance', ascending=False)
print(importance_df)

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      0.431311

BOSQUE ALEATORIO
```

Paso 8: Ejecución del algoritmo de Bosque Aleatorio

8.1. 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)
```

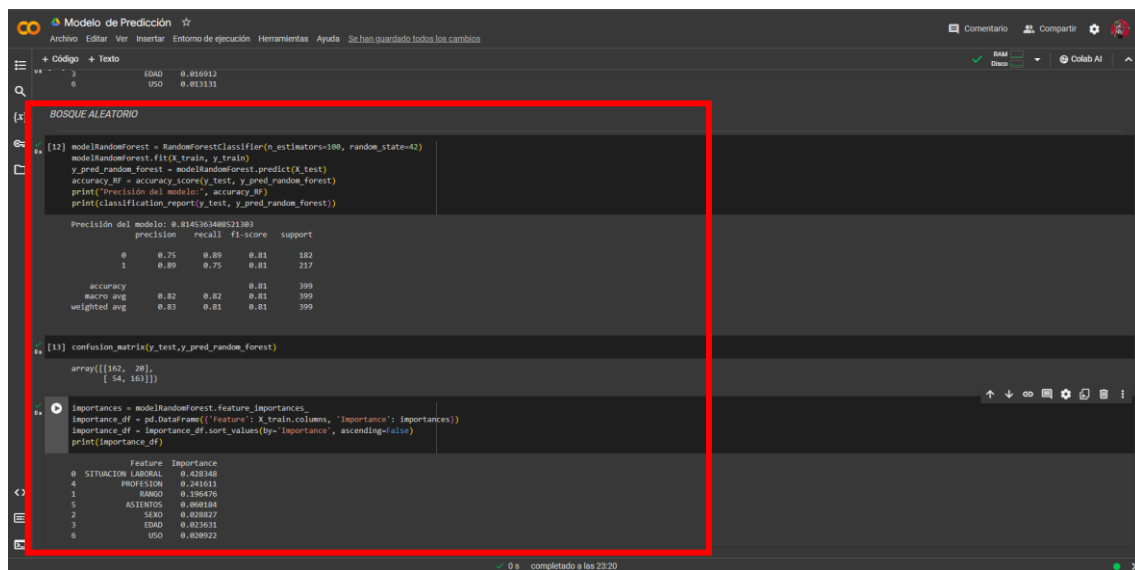
RESULTADO

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]
```

8.2. Ejecución Exitosa



```
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

[11] confusion_matrix(y_test, y_pred_random_forest)

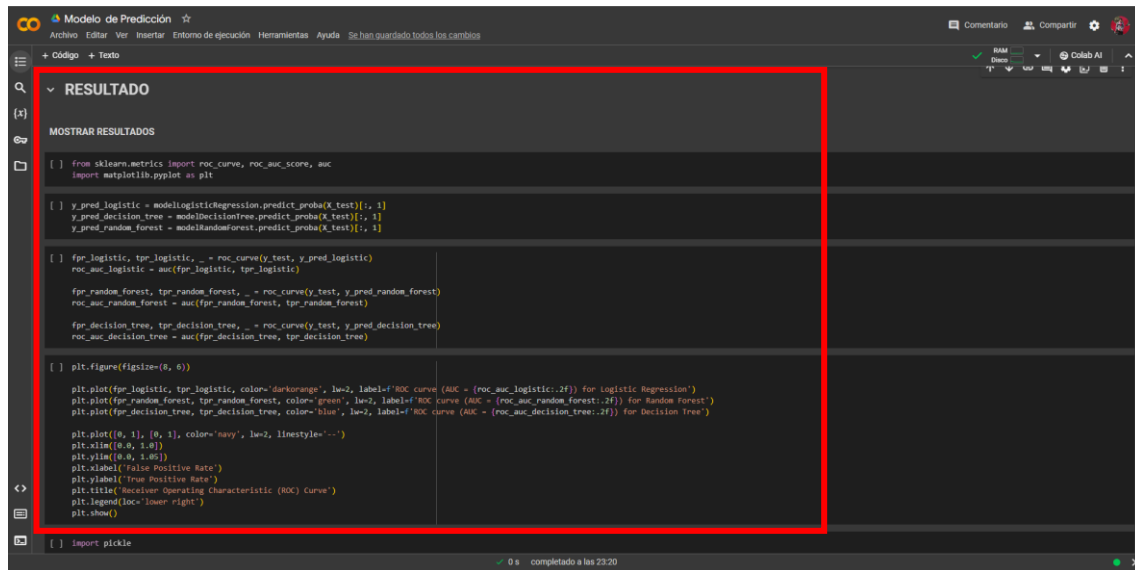
array([[162, 20],
       [ 54, 163]])

[ ] 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

9.1. 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)

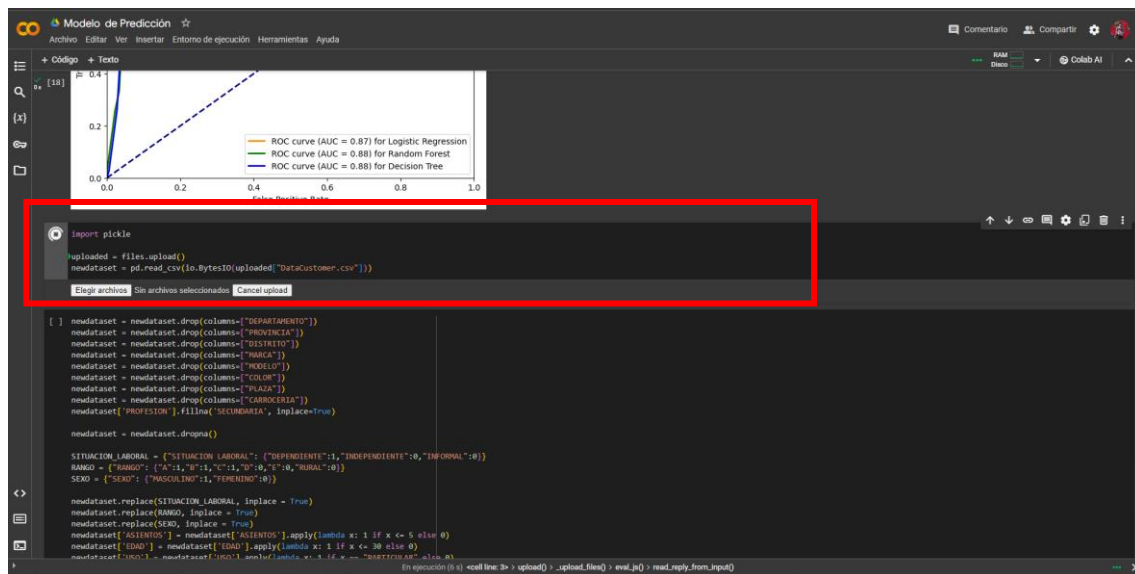
[ ] plt.figure(figsize=(8, 6))

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
```

9.2. Se le pedirá seleccionar un archivo



```
[ ] import pickle

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

Elegir archivos Sin archivos seleccionados Cancel upload

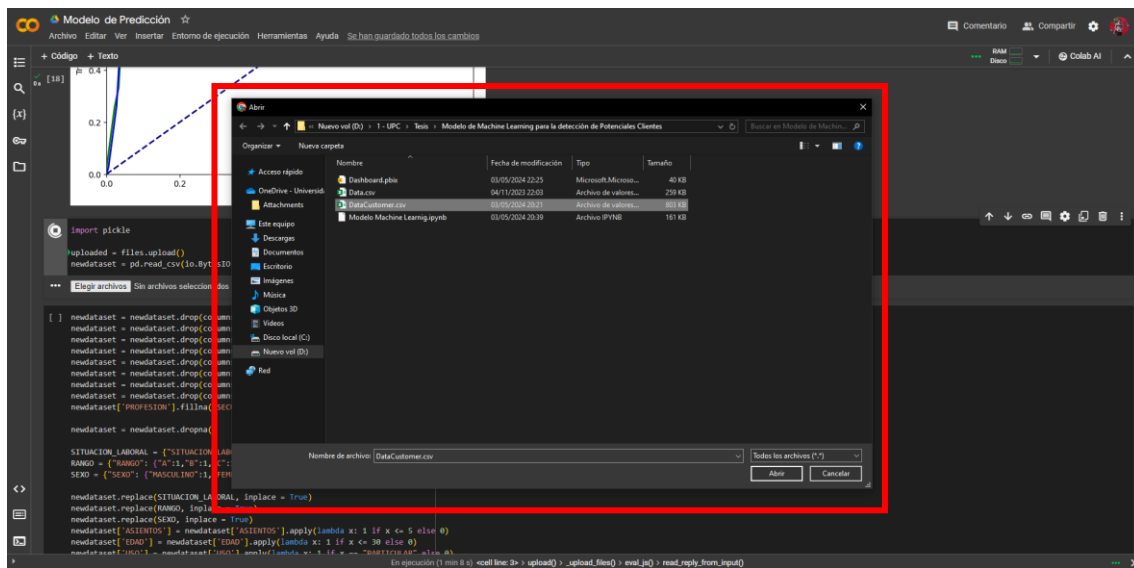
[ ] newdataset = newdataset.drop(columns=["DEPARTAMENTO"])
newdataset = newdataset.drop(columns=["PROVINCIA"])
newdataset = newdataset.drop(columns=["DISTRITO"])
newdataset = newdataset.drop(columns=["MARCA"])
newdataset = newdataset.drop(columns=["MODELO"])
newdataset = newdataset.drop(columns=["COLOR"])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

newdataset = newdataset.dropna()

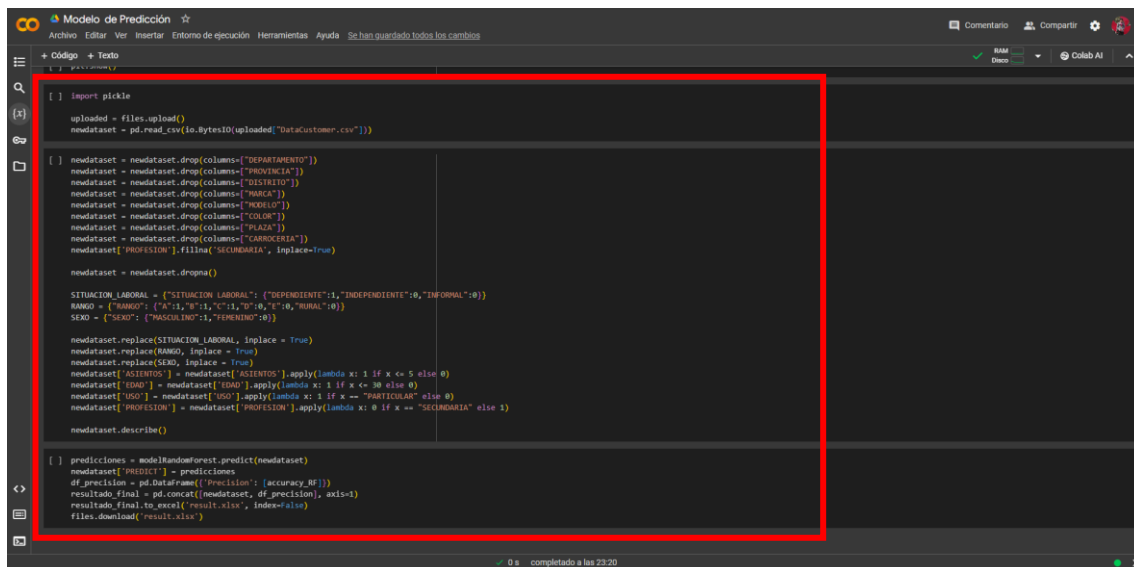
SITUACION_LABORAL = {"SITUACION_LABORAL": {"DEPENDIENTE":1, "INDEPENDIENTE":0, "INFORMAL":0}}
RANGO = {"RANGO": {"A":1, "B":1, "C":1, "D":0, "E":0, "RURAL":0}}
SEXO = {"SEXO": {"MASCULINO":1, "FEMENINO":0}}

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["INDICADOR"] = newdataset["INDICADOR"].apply(lambda x: 1 if x == "Indicador" else 0)
```

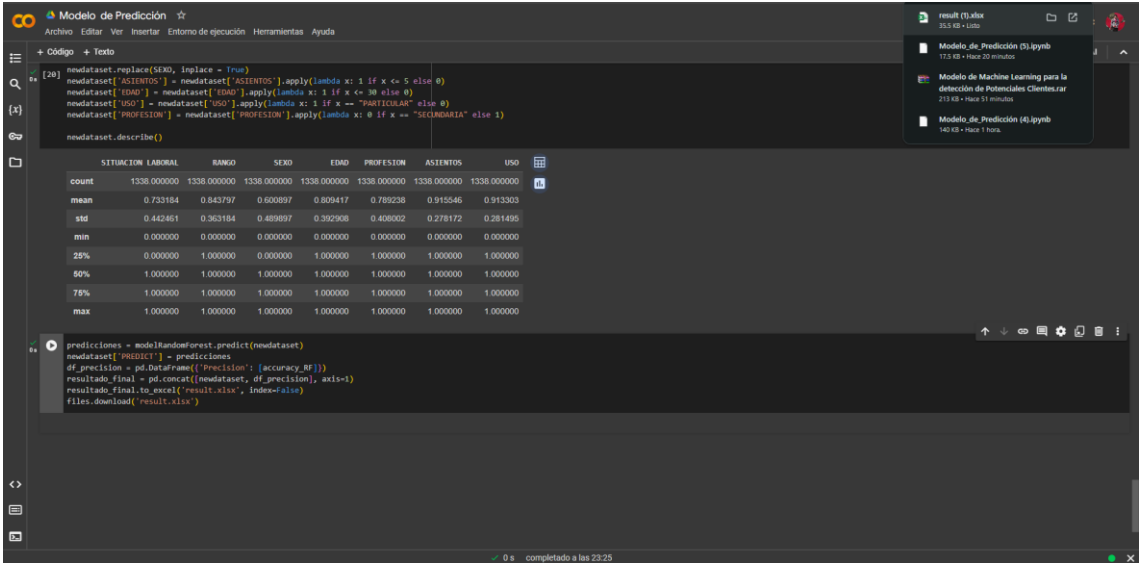
9.3. Subir archivo “DataCustomer.csv”



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



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



```
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)

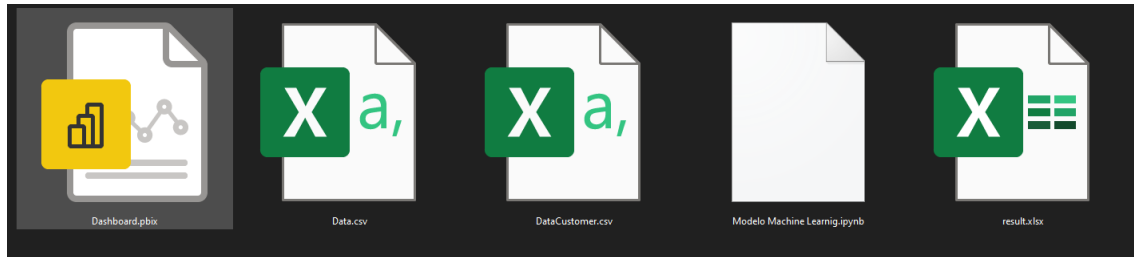
newdataset.describe()
```

	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

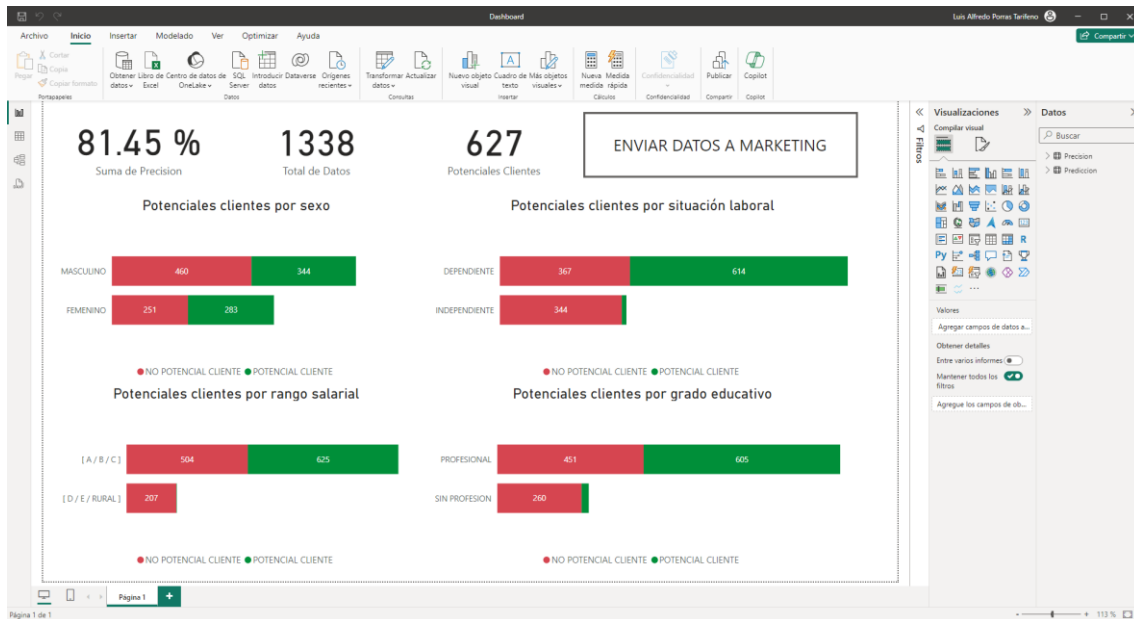
```
predicciones = modelRandomForest.predict(newdataset)
newdataset['PREDICT'] = predicciones
df_precision = pd.DataFrame({'precision': [accuracy_RF]})
resultado_final = pd.concat([newdataset, df_precision], axis=1)
resultado_final.to_excel('result.xlsx', index=False)
files.download('result.xlsx')
```

Paso 10: Cargar los resultados al Power BI

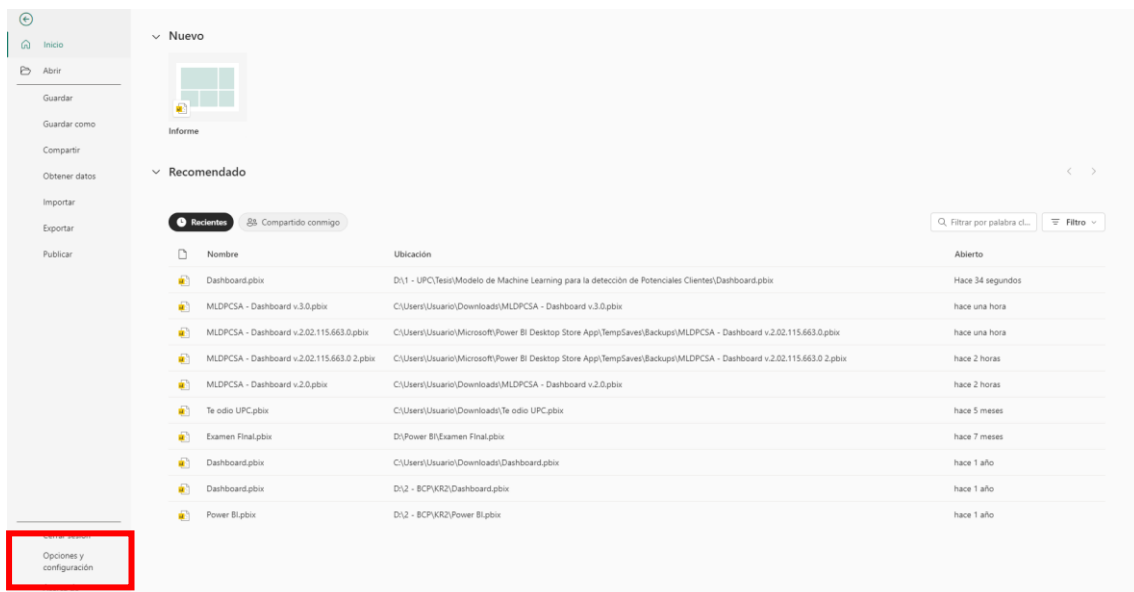
10.1. Abrir el archivo “Dashboard.pbix”



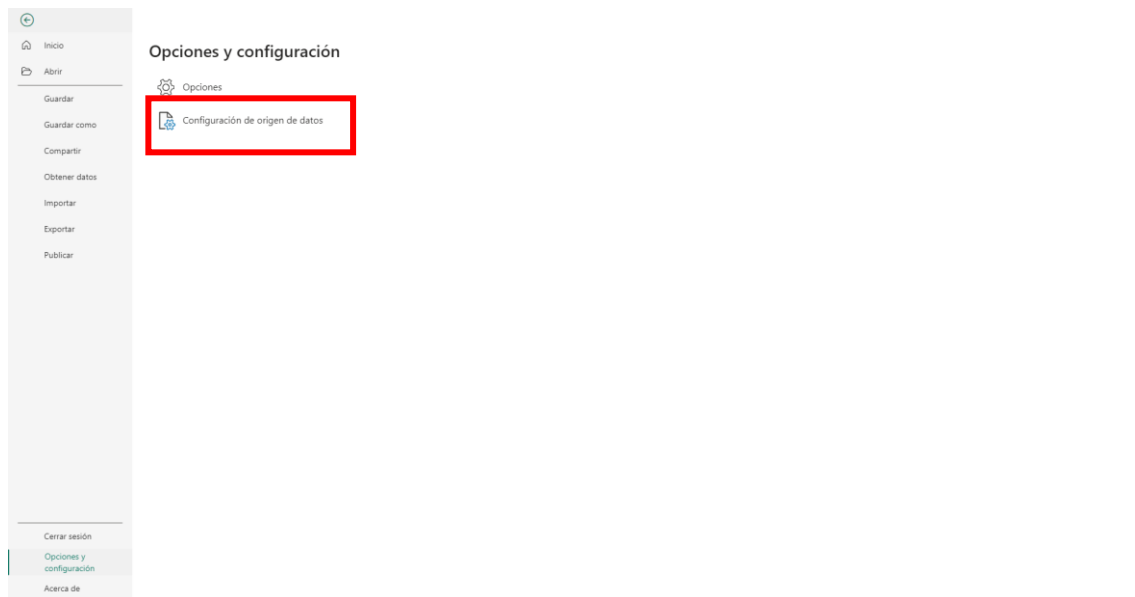
10.2. Se abrirá el dashboard en Power BI



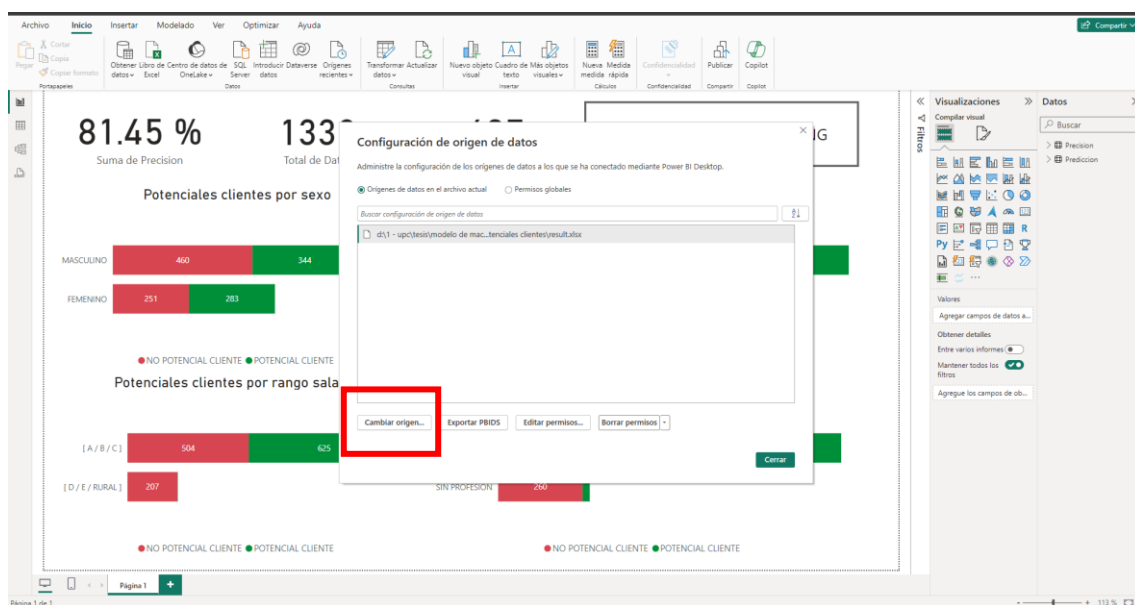
10.3. Se debe seleccionar el nuevo origen de datos. Dar click en “Archivo” > “Opciones y Configuraciones”



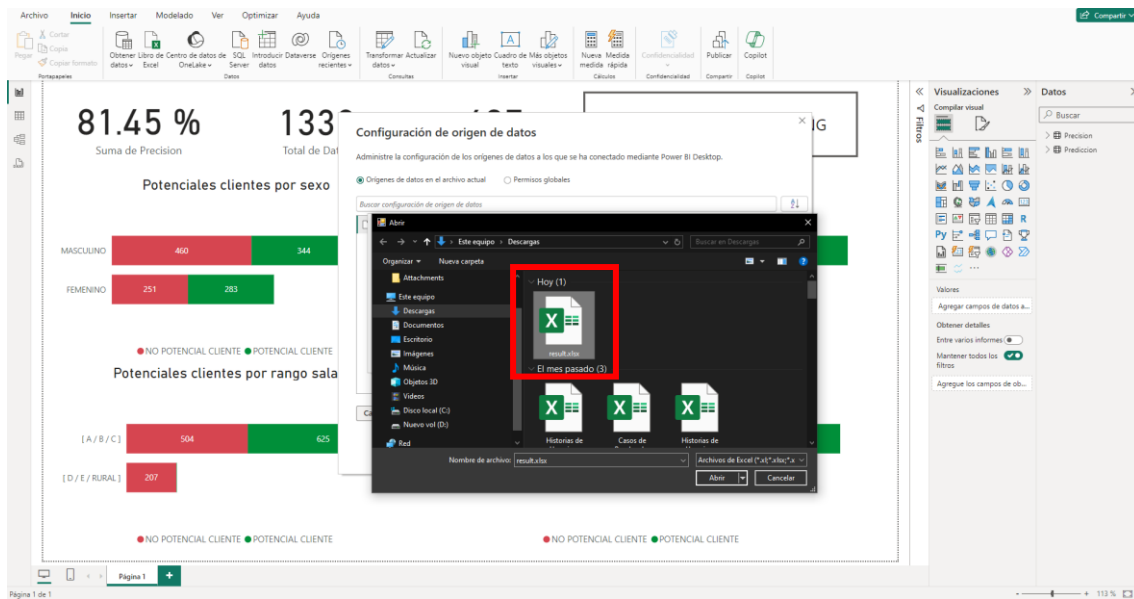
10.4. Dar click en “Configuración de Origen de datos”



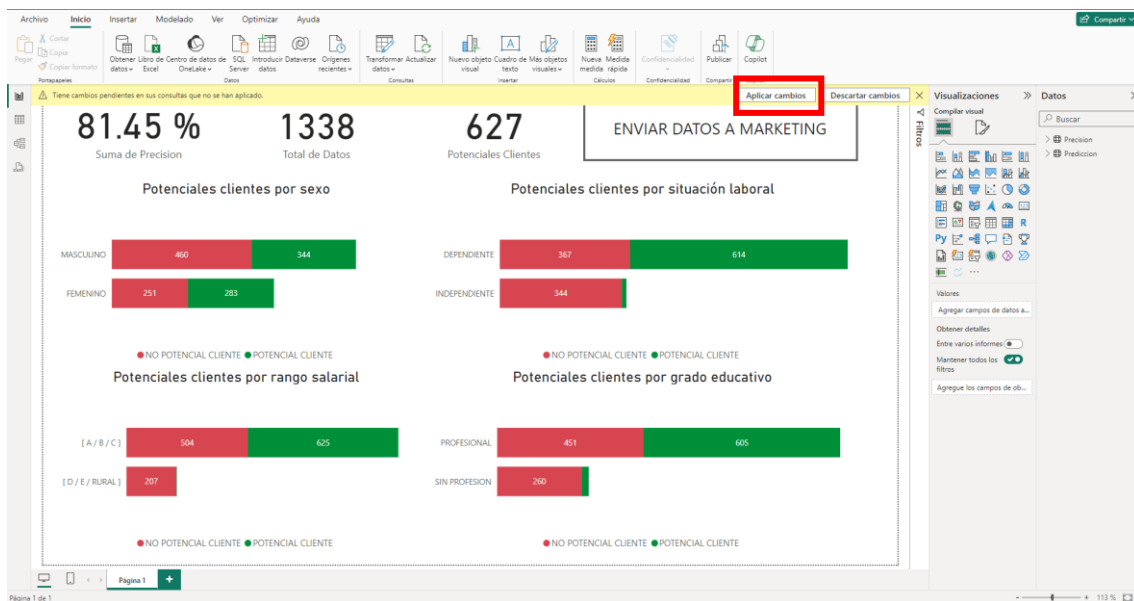
10.5. Dar click en Cambiar origen



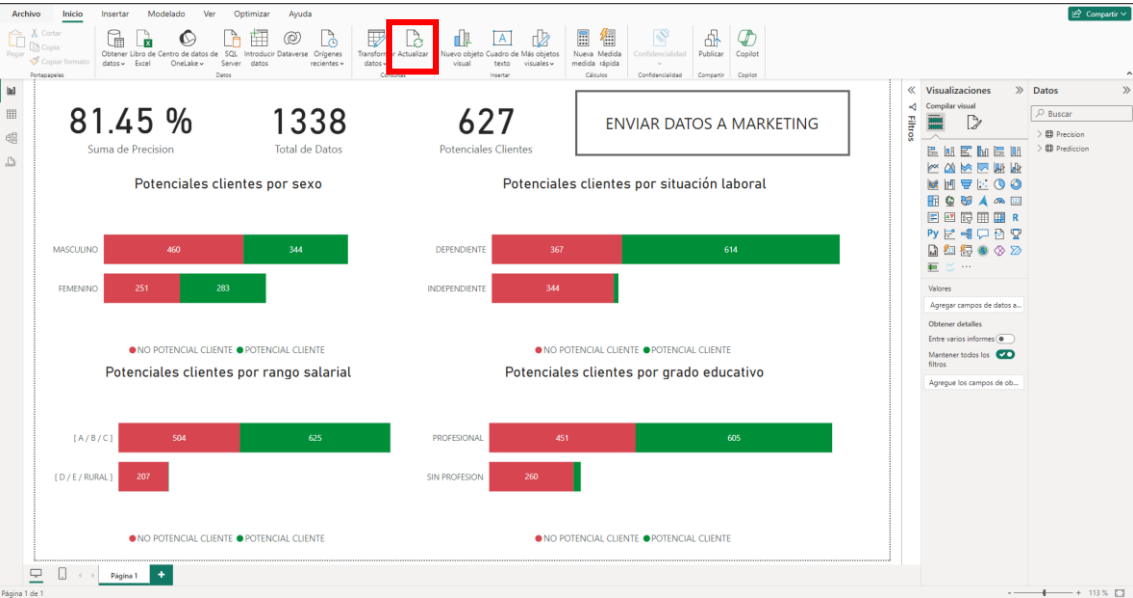
10.6. Seleccionar el archivo “result.xlsx” obtenido del modelo



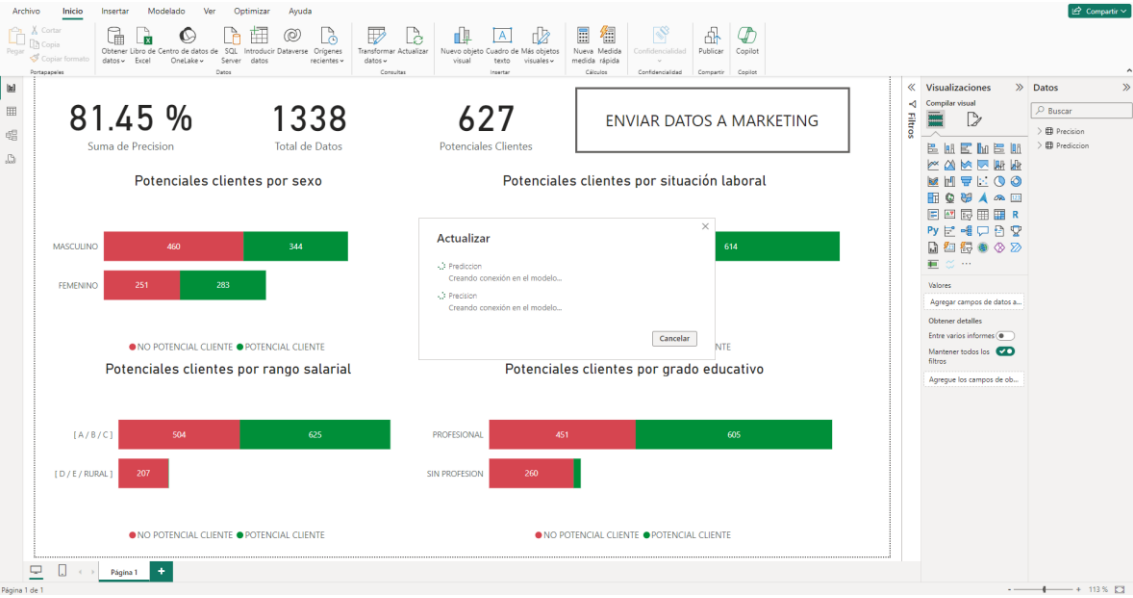
10.7. Dar click en “Aplicar Cambios”



10.8. Dar click en “actualizar”

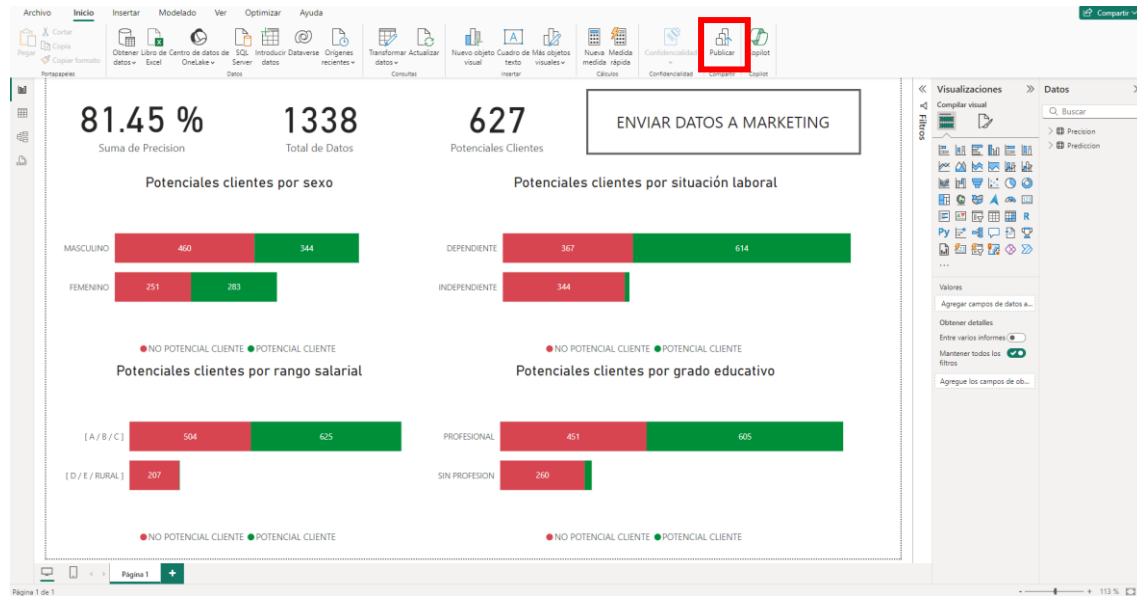


10.9. Carga Exitosa

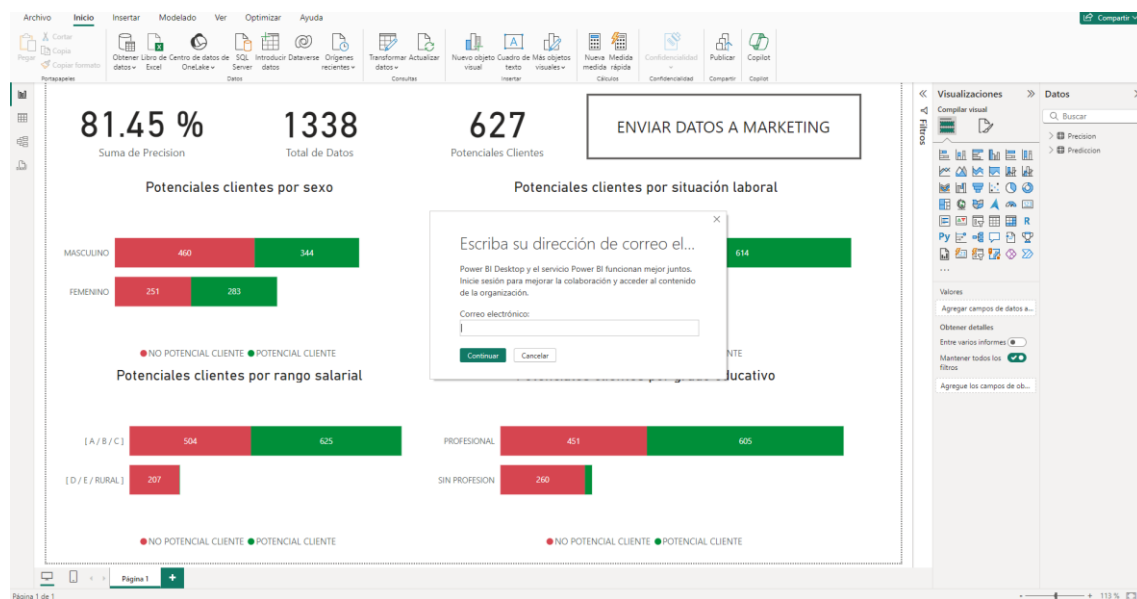


Paso 11 (Opcional): Publicar Dashboard

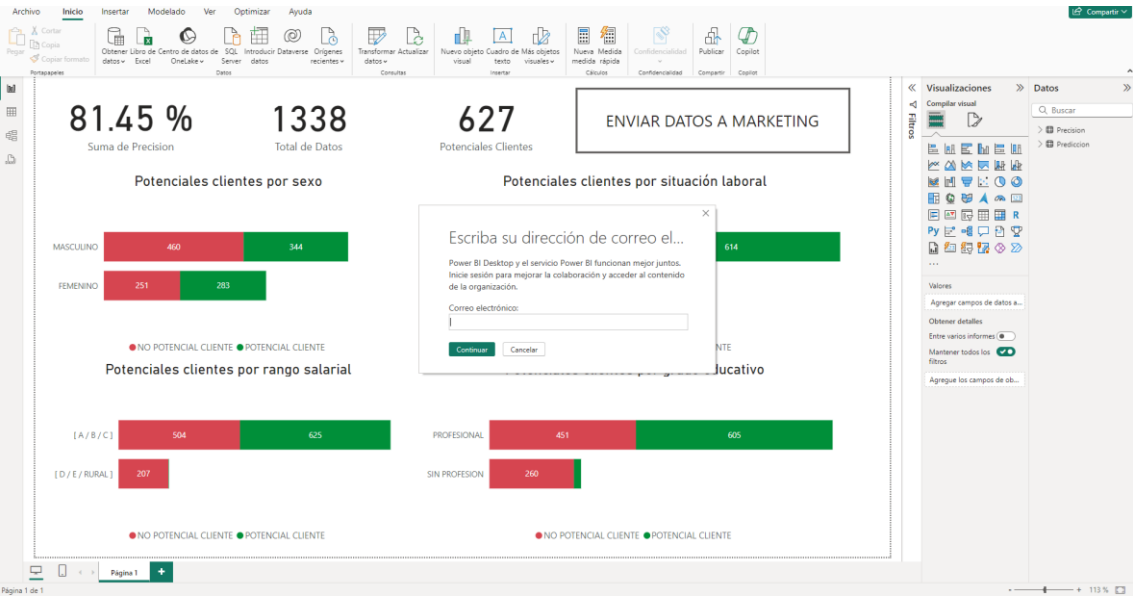
11.1. Click en “Publicar”



11.2. Ingrese su correo con licencia pro o premium



11.3. Ingrese su correo con licencia pro o premium



11.4. Seleccione el workspace y click en “publicar”

