Actividad 13

Jose Manuel Enriquez

29 de marzo del 2025

1 Introduction

1.1 ¿Qué es un árbol de decisión?

Random Forest es un algoritmo de aprendizaje supervisado basado en la combinación de múltiples árboles de decisión para mejorar la precisión y reducir el sobreajuste. Funciona mediante un proceso llamado Bootstrap Aggregating (Bagging), donde se generan subconjuntos de datos aleatorios con reemplazo, y se entrena un árbol de decisión en cada uno. Además, en cada división del árbol, solo se considera un subconjunto aleatorio de características, lo que reduce la correlación entre los árboles y mejora la generalización del modelo. Para la clasificación, el resultado final se obtiene por votación mayoritaria, mientras que en regresión se toma el promedio de las predicciones. Sus principales aplicaciones incluyen la detección de fraudes en finanzas, el diagnóstico de enfermedades en medicina, los sistemas de recomendación en plataformas de contenido, la predicción del clima y la segmentación de imágenes en visión computacional. Entre sus ventajas destacan su capacidad para reducir el sobreajuste, su robustez ante datos ruidosos y su flexibilidad para tareas de clasificación y regresión. Sin embargo, puede ser computacionalmente costoso y menos interpretable que un árbol de decisión individual.

2 Metodología

Este ejercicio fue realizado en una Jupyter Notebook y Python 3.12.4 Lo primero que se hizo fue importar todas las librerías necesarias.

Primero se importan todas las librerias necesarias

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier

from pylab import rcParams

!pip install imbalanced-learn

from imblearn.under_sampling import NearMiss
from imblearn.over_sampling import RandomOverSampler
from imblearn.combine import SMOTETomek
from imblearn.ensemble import BalancedBaggingClassifier

from collections import Counter

#set up graphic style in this case I am using the color scheme from
    xkcd.com
rcParams['figure.figsize'] = 14, 8.7 # Golden Mean
LABELS = ["Normal","Fraud"]
#col_list = ["cerulean","scarlet"]# https://xkcd.com/color/rgb/
#sns.set(style='white', font_scale=1.75, palette=sns.xkcd_palette(
    col_list))

%matplotlib inline
```

Luego importamos los datos

```
df = pd.read_csv("creditcard.csv")
df.head(n=5)
```

Comparamos los datos que tienen 0 y 1

```
pd.value_counts(df['Class'], sort = True) #class comparison 0=
    Normal 1=Fraud
```

Creamos dos dataframes para los dos casos

```
normal_df = df[df.Class == 0] #registros normales
fraud_df = df[df.Class == 1] #casos de fraude
```

Se crean las variables de los datos y hacemos la primera predicción

```
y = df['Class']
X = df.drop('Class', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
    train_size=0.7)

def mostrar_resultados(y_test, pred_y):
    conf_matrix = confusion_matrix(y_test, pred_y)
    plt.figure(figsize=(8, 8))
    sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS
    , annot=True, fmt="d");
    plt.title("Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()
    print (classification_report(y_test, pred_y))
```

```
14
15 def run_model_balanced(X_train, X_test, y_train, y_test):
16     clf = LogisticRegression(C=1.0,penalty='l2',random_state=1,
       solver="newton-cg",class_weight="balanced")
17     clf.fit(X_train, y_train)
18     return clf
19
20 model = run_model_balanced(X_train, X_test, y_train, y_test)
21
22 pred_y = model.predict(X_test)
23 mostrar_resultados(y_test, pred_y)
```

Y se obtuvo la siguiente gráfica:

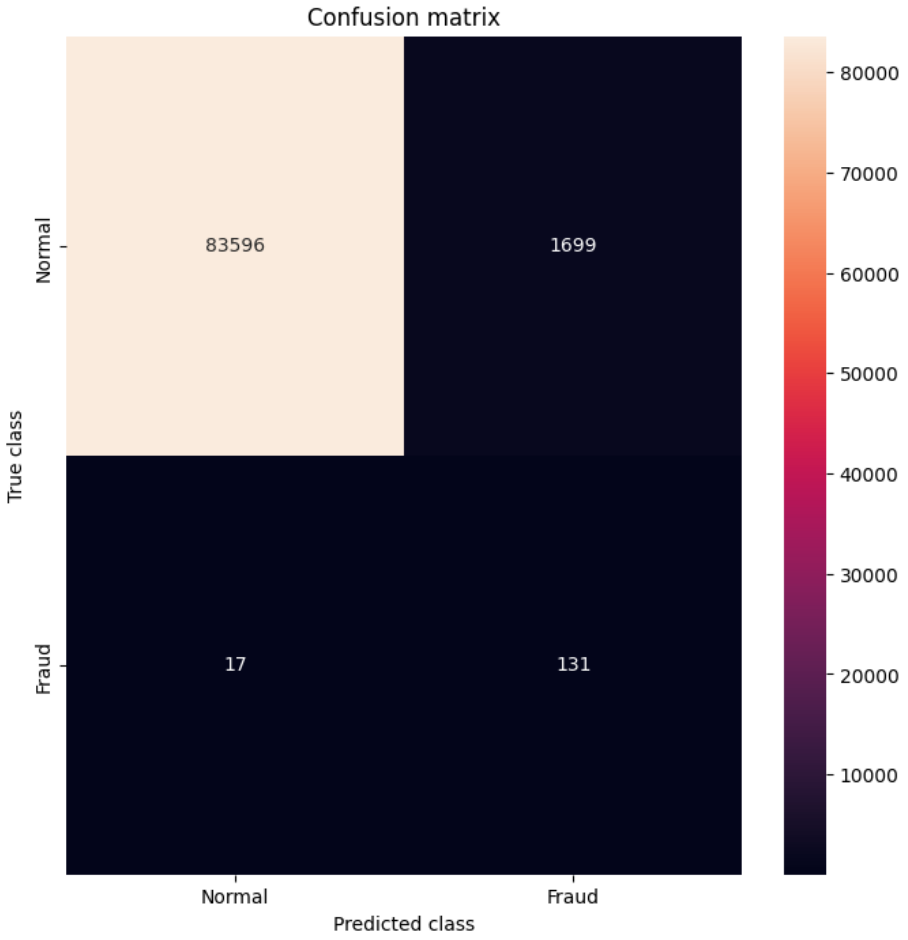

Figure 1: Confussion Matrix 1

Luego se crea un modelo con 100 arboles

```
from sklearn.ensemble import RandomForestClassifier

# Crear el modelo con 100 arboles
model = RandomForestClassifier(n_estimators=100,
                               bootstrap = True,verbose=2,
                               max_features = 'sqrt')
# entrenar!
model.fit(X_train, y_train)

pred_y = model.predict(X_test)
mostrar_resultados(y_test, pred_y)
```

Y ahora se obtuvo la siguiente gráfica

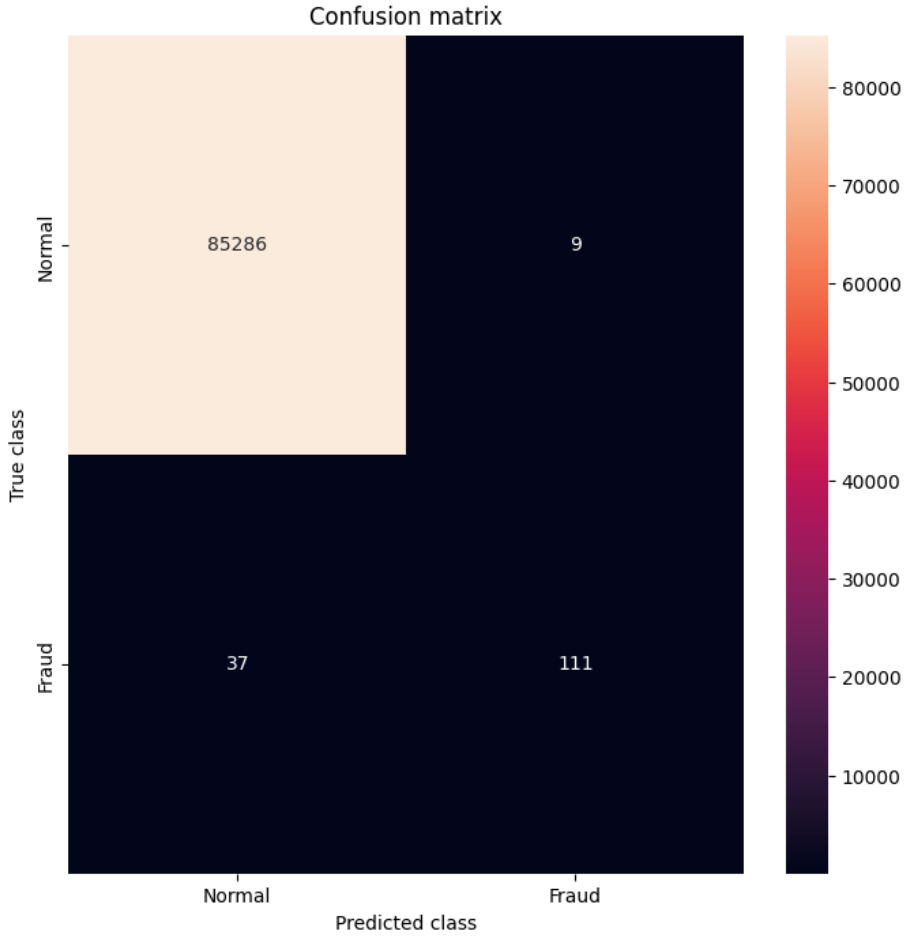

Figure 2: Confussion Matrix 2

3 Resultados

Se obtuvo que la calificación de ROC AUC es de 0.95

4 Conlusiones

Me pareció muy interesante realizar predicciones con random forest con Python ya que es de las primeras veces en las cuales implemento esta herramienta.