Universidad Autónoma de Nuevo León

Licenciatura en Ciencias Computacionales

A13: Programando Random Forest en Python

Materia: Inteligencia Artificial

Maestro: Luis Ángel Gutiérrez Rodríguez

Rebeca Jaramillo Camarillo

Matrícula: 2132988

Grupo: 031

Fecha: 30 de marzo de 2025

1. Introducción

1.1. ¿Qué es Random Forest?

Un modelo Random Forest está formado por múltiples árboles de decisión individuales. Cada uno de estos árboles es entrenado con una muestra ligeramente diferente de los datos de entrenamiento, generada mediante una técnica conocida como bootstrapping. Para realizar predicciones sobre nuevas observaciones, se combinan las predicciones de todos los árboles que conforman el modelo. En cada árbol individual, las observaciones se distribuyen a través de bifurcaciones (nodos), dando forma a la estructura del árbol hasta llegar a un nodo terminal. La predicción de una nueva observación se obtiene al agregar las predicciones de todos los árboles individuales que conforman el modelo.

Muchos métodos predictivos generan modelos globales en los que una única ecuación se aplica a todo el espacio muestral. Cuando el caso de uso implica múltiples predictores, que interaccionan entre ellos de forma compleja y no lineal, es muy difícil encontrar un único modelo global que sea capaz de reflejar la relación entre las variables. Los métodos estadísticos y de machine learning basados en árboles engloban a un conjunto de técnicas supervisadas no paramétricas que consiguen segmentar el espacio de los predictores en regiones simples, dentro de las cuales es más sencillo manejar las interacciones. Es esta característica la que les proporciona gran parte de su potencial.

2. Metodología

A continuación, se muestran los pasos para resolver el ejercicio:

Importar librerías necesarias

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

from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA

from pylab import rcParams
```

2.1. Cargar los datos de entrada

```
df = pd.read_csv("creditcard.csv")
print(df.head())
print(df.shape)
print(pd.value_counts(df['Class'], sort=True))
```

2.2. Visualización del desbalance de clases

```
count_classes = pd.value_counts(df['Class'], sort=True)
count_classes.plot(kind='bar', rot=0)
plt.xticks(range(2), LABELS)
plt.title("Frecuencia por numero de observaciones")
plt.xlabel("Clase")
plt.ylabel("Numero de Observaciones")
plt.show()
```

2.3. Preparar los datos

```
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,
    random_state=42)

# Reducir el conjunto de datos a 2 dimensiones para visualizarlos
pca = PCA(n_components=2)
pca.fit(X)
X_reduced = pca.transform(X)

fig, ax = plt.subplots(1, 2, figsize=(15, 5))

ax[0].scatter(X_reduced[y == 0, 0], X_reduced[y == 0, 1], label="Normal",
    alpha=0.2)
ax[0].scatter(X_reduced[y == 1, 0], X_reduced[y == 1, 1], label="Fraude",
    alpha=0.2)
ax[0].set_title('PCA del dataset original')
ax[0].legend()

sns.countplot(x=y, ax=ax[1])
ax[1].set_title('Numero de observaciones por clase')
plt.show()
```

2.4. Modelo sin balancear

```
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("Matriz de confusion")
    plt.ylabel('Clase verdadera')
    plt.xlabel('Clase predicha')
    plt.show()
    print(classification_report(y_test, pred_y))
```

2.5. Crear modelo y entrenarlo

```
model = RandomForestClassifier(
    n_estimators=100,
    bootstrap=True,
    verbose=2,
    max_features='sqrt'
)

print("Entrenando el modelo Random Forest...")
model.fit(X_train, y_train)

print("\nEvaluando el modelo Random Forest...")
pred_y_rf = model_rf.predict(X_test)
mostrar_resultados(y_test, pred_y_rf)
```

3. Resultados

Desbalance de clases

Se observa un desbalance en los datos, donde solo 492 registros pertenecen a los casos de fraude.

Class	
0	284315
1	492

Figura 1: Comparación de clases

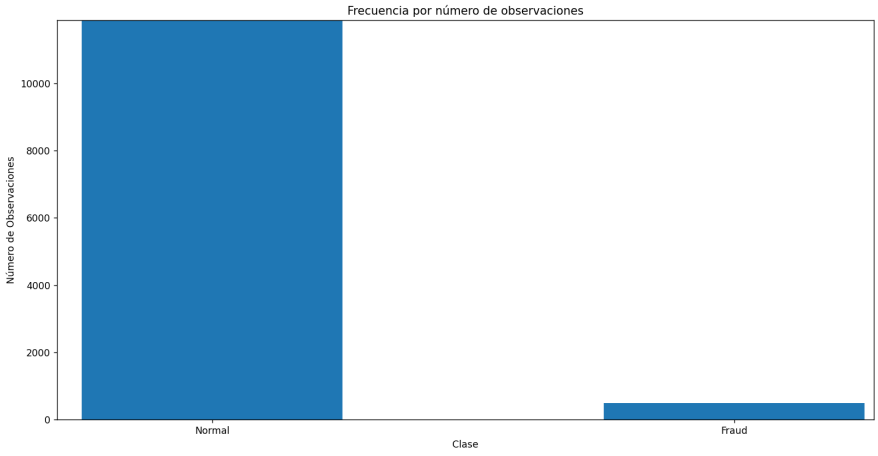

Figura 2: Gráfica de comparación de clases

Matriz de confusión

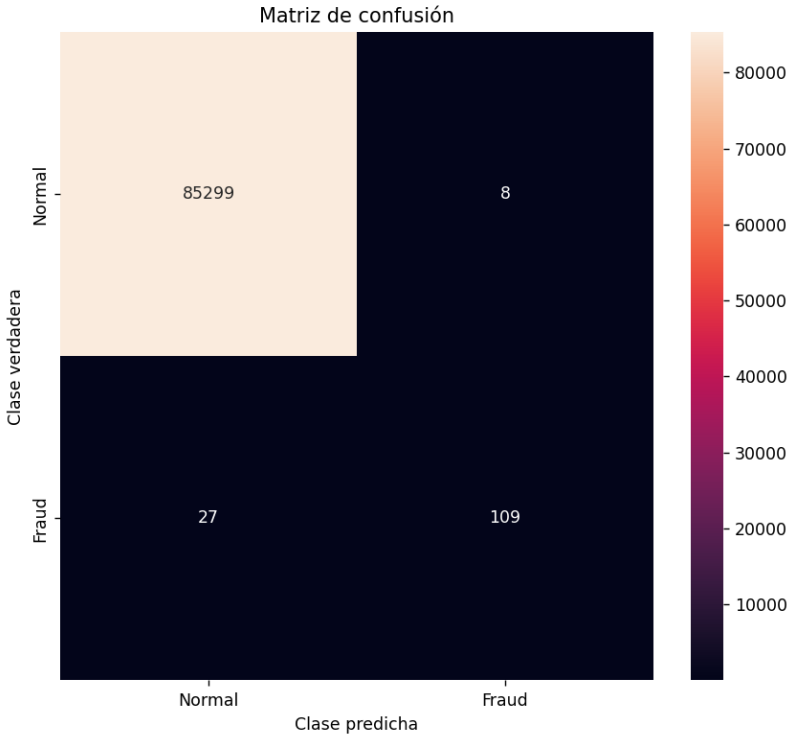

Figura 3: Matriz de confusión

Métricas sobre el conjunto de test

	precision	recall	f1-score	support
0	1.00	1.00	1.00	85307
1	0.93	0.80	0.86	136
accuracy			1.00	85443
macro avg	0.97	0.90	0.93	85443
weighted avg	1.00	1.00	1.00	85443

Figura 4: Métricas

Los resultados clasificados con error serían 8+27 muestras. Ademas, los casos de fraude tienen un valor de 0.93, es decir, el modelo detecta la mayoria de los fraudes.

4. Conclusión

En esta actividad se analizo un conjunto de datos de transacciones para generar un Random Forest que pueda identificar patrones de fraude. A pesar del desbalance extremo en los datos, el modelo logró un alto recall (sensibilidad), lo que significa que detectó la mayoría de los casos de fraude.