Random Forest en Python

1 Introducción

Random Forest es un método de aprendizaje supervisado que combina múltiples árboles de decisión entrenados con subconjuntos aleatorios de datos y características. La predicción final resulta del voto mayoritario (clasificación) o promedio (regresión) de los árboles individuales.

2 Metodología

Para realizar el ejercicio de regresión logística, tuve que seguir los siguientes pasos: Análisis de los datos, Creación y Ajuste del modelo Random Forest, Evaluación del Modelo y Comparación con Regresión Logística.

```
#Importamos las librerias
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

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

rcParams['figure.figsize'] = 14, 8.7
LABELS = ["Normal","Fraud"]
%matplotlib inline

#Creamos un dataframe
df = pd.read_csv("creditcard.csv")
```

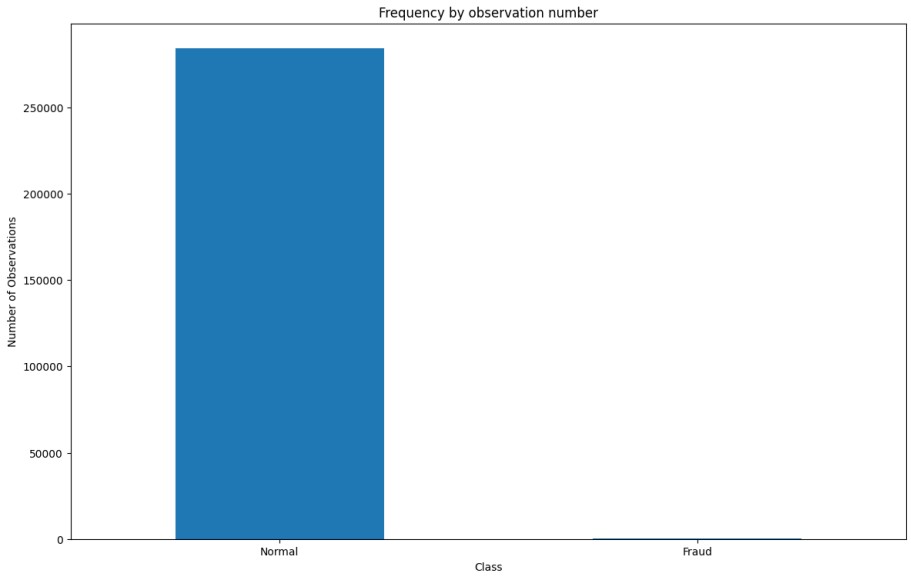
2.1 Análisis de los datos

En esta sección vamos a visualizar el desbalanceo de los datos en el *dataset* que se está usando.

```
print(pd.Series(df['Class'].value_counts(sort = True)))

count_classes = pd.Series(df['Class'].value_counts(sort = True))
count_classes.plot(kind = 'bar', rot=0)
plt.xticks(range(2), LABELS)
plt.title("Frequency by observation number")
plt.xlabel("Class")
plt.ylabel("Number of Observations")
```

```
0    284315
1       492
Name: Class, dtype: int64
```



Podemos ver que existe una gran diferencia en el número de observaciones de tarjetas de crédito normales y casos de fraude.

2.2 Creación y Ajuste del modelo

En esta sección vamos a separar los datos en conjuntos de entrenamiento y prueba para ajustar el modelo de regresión Random Forest.

```
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)

from sklearn.ensemble import RandomForestClassifier

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

model.fit(X_train, y_train)
```

```
building tree 1 of 100
building tree 2 of 100
building tree 3 of 100
...
building tree 98 of 100
building tree 99 of 100
building tree 100 of 100
```

No siempre es la mejor opción usar la mayor cantidad de árboles estimadores en el Random Forest, la cantidad de estimadores va a depender mucho del *dataset*.

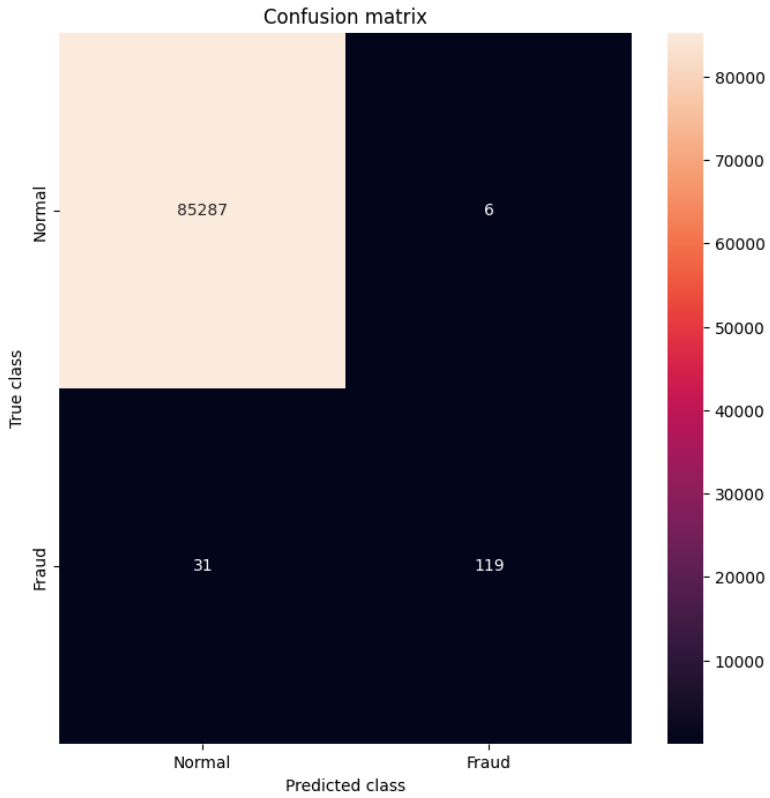
2.3 Evaluación del modelo

En esta sección vamos a realizar las predicciones del modelo y graficar la matriz de confusión. Esta matriz nos va a mostrar la diferencia entre la clase predecida y la clase real de cada observación del conjunto de prueba.

```
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))

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



	precision	recall	f1-score	support
0	1.00	1.00	1.00	85293
1	0.95	0.79	0.87	150
accuracy			1.00	85443
macro avg	0.98	0.90	0.93	85443
weighted avg	1.00	1.00	1.00	85443

La gran mayoría de los datos fueron asignados a su clase real, con solo 37 casos en los que el modelo se equivocó.

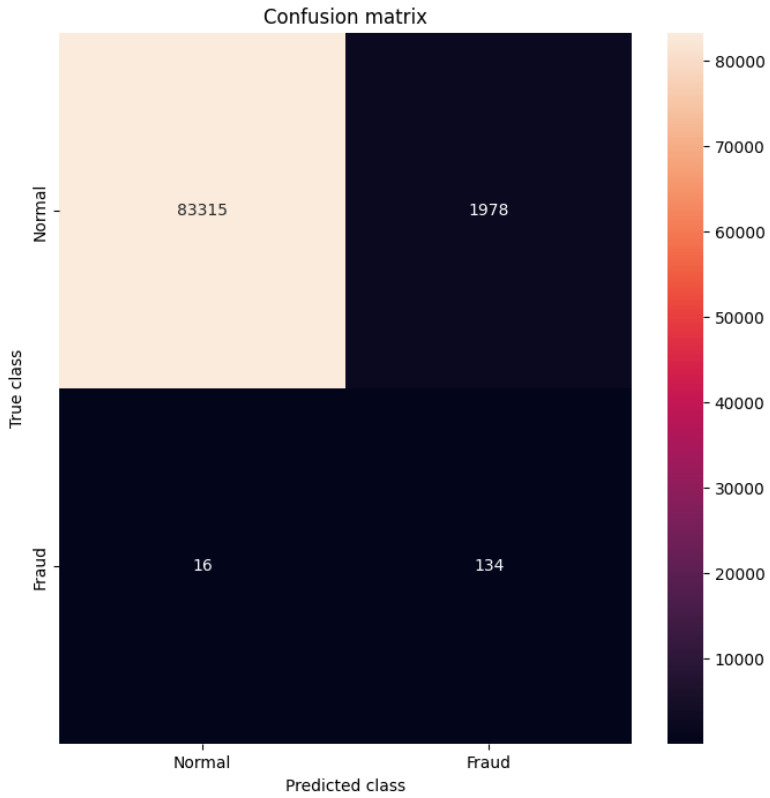
3 Resultados

Otra cosa que podemos hacer es comparar los resultados de este modelo con un modelo de regresión logística para comparar sus resultados y ver cual es más preciso con estos datos.

```
def run_model_balanced(X_train, X_test, y_train, y_test):
    clf = LogisticRegression(C=1.0,penalty='l2',random_state=1,
        solver="newton-cg",class_weight="balanced")
    clf.fit(X_train, y_train)
    return clf

model = run_model_balanced(X_train, X_test, y_train, y_test)
```

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



	precision	recall	f1-score	support
0	1.00	0.98	0.99	85293
1	0.06	0.89	0.12	150
accuracy			0.98	85443
macro avg	0.53	0.94	0.55	85443
weighted avg	1.00	0.98	0.99	85443

Como se puede ver, el modelo de regresión logística solo reconoció el 6% de los casos de fraude con tarjetas de crédito. Podemos confirmar que el modelo Random Forest es una mejor opción con estos datos y con casos similares.

4 Conclusión

Los modelos de Random Forest son herramientas del aprendizaje supervisado que hacen uso de varios árboles de decisión que toman decisiones independientemente de cada uno y donde el modelo escoge el resultado que se predijo más veces en el caso de variables dependientes categóricas, o el promedio de todos los resultados para los valores númericos.

Como se vió en este caso, el modelo Random Forest tuvo un mejor desempeño en predecir los casos de fraude que el modelo de regresión logística, pero también hay que tomar en cuenta que los modelos Random Forest se tardan más en entrenar, por lo que la decisión de usar un Random Forest o un modelo de regresión lineal o logístico va a depender de las necesidades de los clientes.