

Actividad 13: Programando Random Forest en Python

García Herrera Carlos Eduardo Marzo 2025

1 Introducción

Es un algoritmo de aprendizaje automático basado en el uso de múltiples árboles de decisión para mejorar la precisión y reducir el sobreajuste. Se considera un método de ensamble, ya que combina varios árboles de decisión entrenados de manera independiente y luego promedia sus resultados (para regresión) o usa votación mayoritaria (para clasificación).

Este algoritmo introduce aleatoriedad tanto en la selección de datos como en la elección de características para cada árbol, lo que le permite ser más robusto y generalizable en comparación con un solo árbol de decisión. Se utiliza ampliamente en tareas de clasificación, regresión y selección de características en áreas como el análisis financiero, la medicina y el reconocimiento de patrones.

2 Metodología

2.1 Parte 1: Creación del Ambiente Virtual

```
#Automatic creation of an virtual environment to run the script and install
the libraries
import subprocess
import os
import venv
import sys
script_dir = os.path.dirname(os.path.realpath(__file__))
env_name = os.path.join(script_dir, "VirtualEnv")
if os.path.exists(os.path.join(script_dir, "VirtualEnv")):
    #Checks if the VirtualEnv is activated (This is the path to the Python
    installation currently in use. If the virtual environment is active,
    sys.prefix will point to the virtual environment directory, while
    sys.base_prefix points to the global Python installation.)
    if sys.prefix == sys.base_prefix:
        print("Activating the Virtual Environment...")
        python_exe = os.path.join(env_name, "Scripts", "python")
        subprocess.run([python_exe, __file__])
    else:
        print("Installing the Required Libraries on a New Virtual Environment")
```

```

venv.create(env_name, with_pip=True)

# Step 2: Install the libraries
libraries = ["scikit-learn", "matplotlib", "seaborn", "pandas", "numpy"]
for lib in libraries:
    subprocess.run([os.path.join(env_name, "Scripts", "pip"), "install",
lib], check=True)

#Re-Run the script with the Virtual Env Activated
python_exe = os.path.join(env_name, "Scripts", "python")
subprocess.run([python_exe, __file__])

```

2.2 Parte 2: Importación de las librerías Necesarias

```

#Random Forest
#Importacion de 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 sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score

from matplotlib import rcParams

from collections import Counter

```

2.3 Parte 3: Carga de los datos de Entrada y análisis de los registros

```

#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"]
df = pd.read_csv("creditcard.csv")
pd.set_option('display.max_columns', None) #Muestra todas las columnas del
dataSet

print("Primeros 5 Registros")

```

```

print(df.head())
print("\n\nForma del dataset")
print(df.shape)

print("Desbalanceo en el DataSet")
print(pd.value_counts(df['Class'], sort = True))

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

#Creacion del DataSet
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)

```

2.4 Parte 4: Funciones para Mostrar la matriz de confusión

```

#Cracion de una Matriz de Confusion
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))

#Ejecucion del Modelo con Regresion Logistica para Comparacion
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

```

2.5 Parte 5: Regresión logística

```

#Modelo Regresion Logistica
model = run_model_balanced(X_train, X_test, y_train, y_test)

#Mostrar Resultados
pred_y = model.predict(X_test)
print("\n\nResultados Regresion Logistica")

```

```
mostrar_resultados(y_test, pred_y)
```

2.6 Parte 6: Random Forest 1

```
#Creacion del RandomForest1
# Crear el modelo con 100 arboles
model = RandomForestClassifier(n_estimators=100,
                              bootstrap = True, verbose=2,
                              max_features = 'sqrt',
                              n_jobs=-1) #USA TODOS LOS NUCLEOS DISPONIBLES

# entrenar!
model.fit(X_train, y_train)

#Mostrar Resultados
pred_y = model.predict(X_test)
print("\n\nResultados Random Forest 1")
mostrar_resultados(y_test, pred_y)
```

2.7 Parte 7: Random Forest 2

```
# Creacion del RandomForest2 (Cambio de Parametros)
model = RandomForestClassifier(n_estimators=100, class_weight="balanced",
                              max_features = 'sqrt', verbose=2,
                              max_depth=6,
                              oob_score=True, random_state=50,
                              n_jobs=-1) #USA TODOS LOS NUCLEOS DISPONIBLES

# a entrenar
model.fit(X_train, y_train)

#Mostrar Resultados
pred_y = model.predict(X_test)
print("\n\nResultados Random Forest 2")
mostrar_resultados(y_test, pred_y)
```

2.8 Parte 8: Calculo de ROC AUC

```
# Calculate roc auc
roc_value = roc_auc_score(y_test, pred_y)
print("\n\nROC AUC Value:")
print(roc_value)

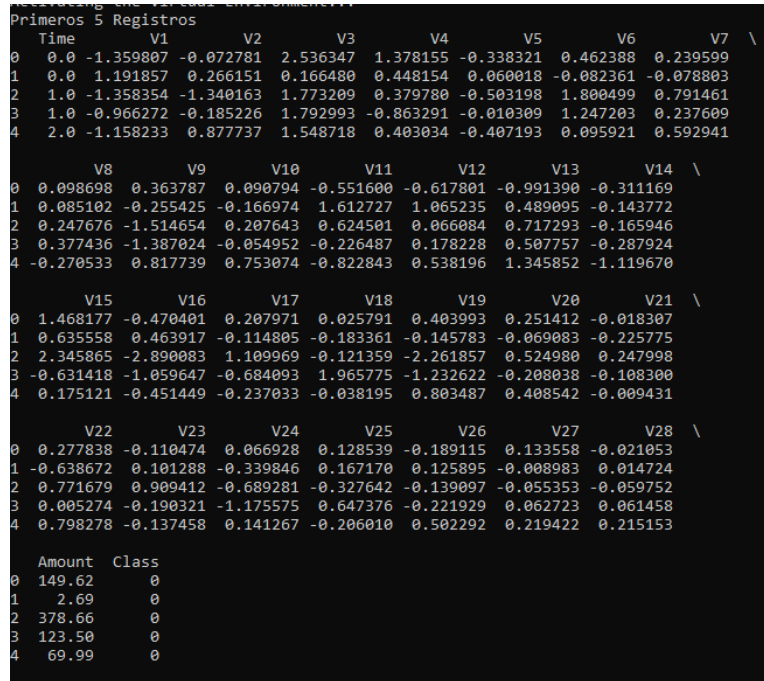
input("Presiona Cualquier Tecla para Continuar...")
```

3 Resultados

Al ejecutar el script de Python la información obtenida es la siguiente, cabe recalcar que la información es obtenida de forma secuencial a como se muestran a continuación:

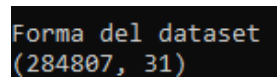
3.1 Vista de los Datos

Se hace un análisis de los datos al ver distintas características de ellos, por medio de distintas graficas de barras para poder reconocer ciertas relaciones entre ellos.



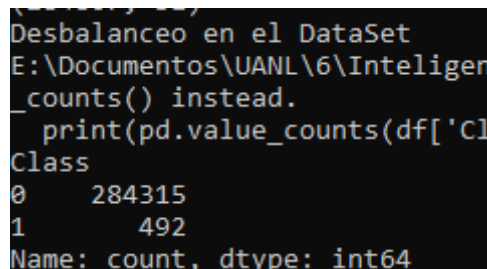
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169	1.468177	-0.470401	0.207971	0.025791	0.403993	0.251412	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772	0.635558	0.463917	-0.114805	-0.183361	-0.145783	-0.069083	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946	2.345865	-2.890083	1.109969	-0.121359	-2.261857	0.524980	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924	-0.631418	-1.059647	0.684093	1.965775	-1.232622	-0.208038	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670	0.175121	-0.451449	-0.237033	-0.038195	0.803487	0.408542	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

Ilustración 1: Vista preliminar de los datos, tanto la forma como las columnas del DataSet



```
Forma del dataset  
(284807, 31)
```

Ilustración 2: Forma del DataSet



```
Desbalanceo en el DataSet  
E:\Documentos\UANL\6\Inteligencia Artificial\Datos\dataset.csv  
_counts() instead.  
print(pd.value_counts(df['Class']))  
Class  
0    284315  
1     492  
Name: count, dtype: int64
```

Ilustración 3: Clasificación por Tipo de transacción

3.2 Regresión Logística

Primero se hizo una regresión logística con el objetivo de comparar los resultados obtenidos por medio del Random Forest.

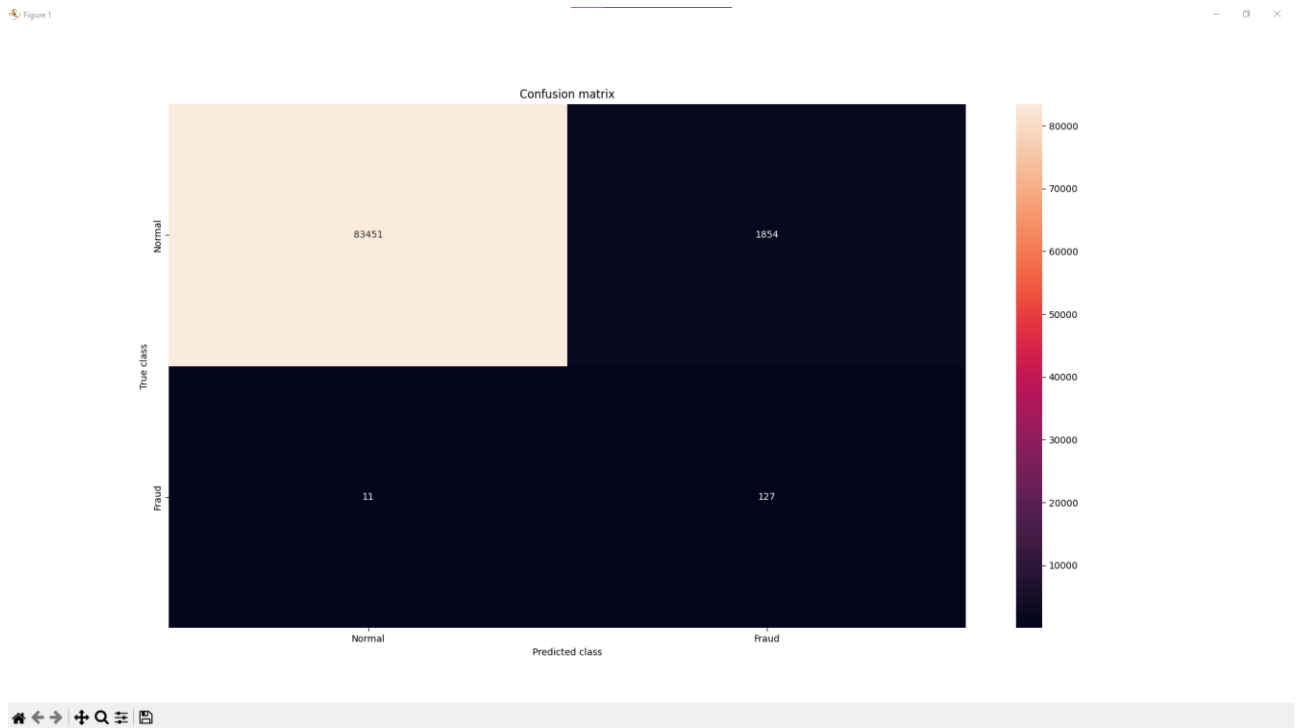


Ilustración 4: Matriz de confusión de la regresión logística

Resultados Regresion Logistica				
	precision	recall	f1-score	support
0	1.00	0.98	0.99	85305
1	0.06	0.92	0.12	138
accuracy			0.98	85443
macro avg	0.53	0.95	0.55	85443
weighted avg	1.00	0.98	0.99	85443

Ilustración 5: Resultados del Modelo calculado por medio de la regresión Logistica

3.3 Random Forest 1

El primer modelo de Random Forest calculado obtuvo los resultados siguientes:

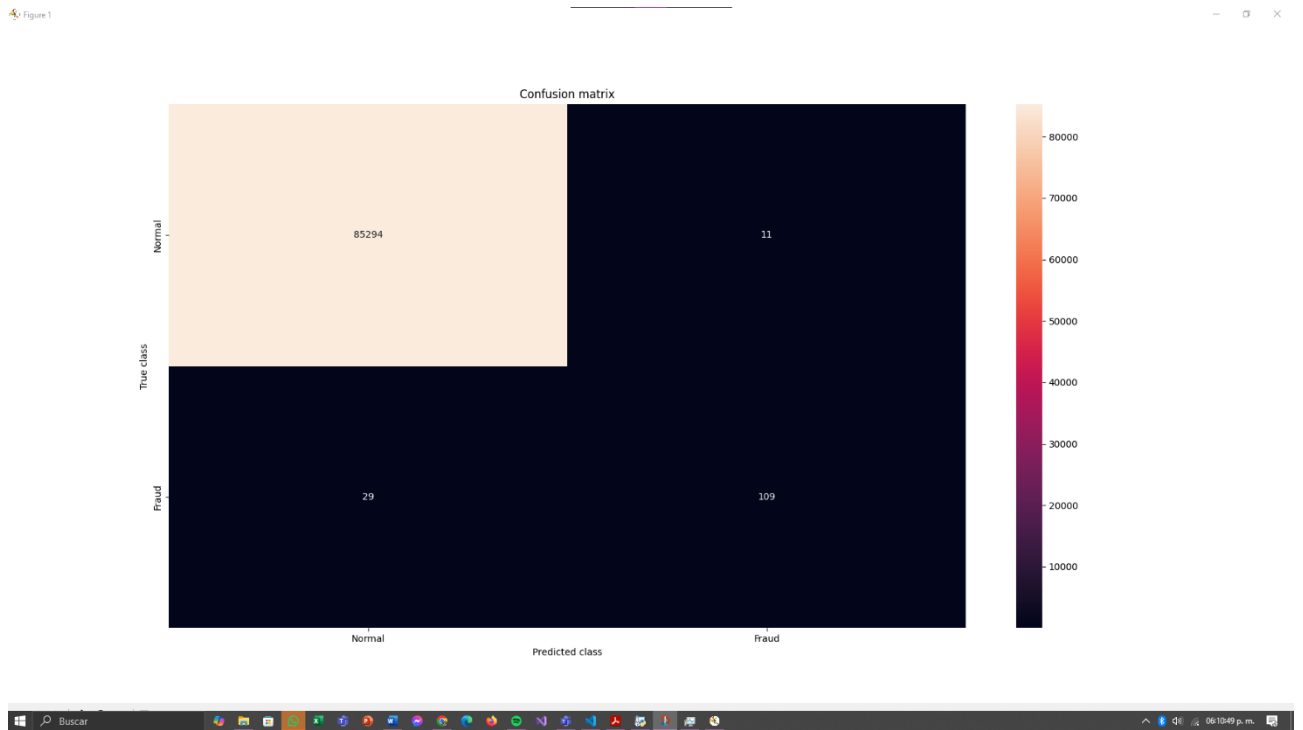


Ilustración 6: Matriz de confusión de Random Forest 1

Resultados Random Forest 1					
	precision	recall	f1-score	support	
0	1.00	1.00	1.00	85305	
1	0.91	0.79	0.84	138	
accuracy			1.00	85443	
macro avg	0.95	0.89	0.92	85443	
weighted avg	1.00	1.00	1.00	85443	

Ilustración 7: Resultados del Modelo calculado por medio de Random Forest 1

3.4 Random Forest 2

El segundo modelo de Random Forest, se creó con el propósito de que el proceso de entrenamiento sea más rápido, aunque no se notó gran diferencia, debido a que se usaron todos los núcleos del procesador para entrenar cada uno de los modelos, los resultados son los siguientes:

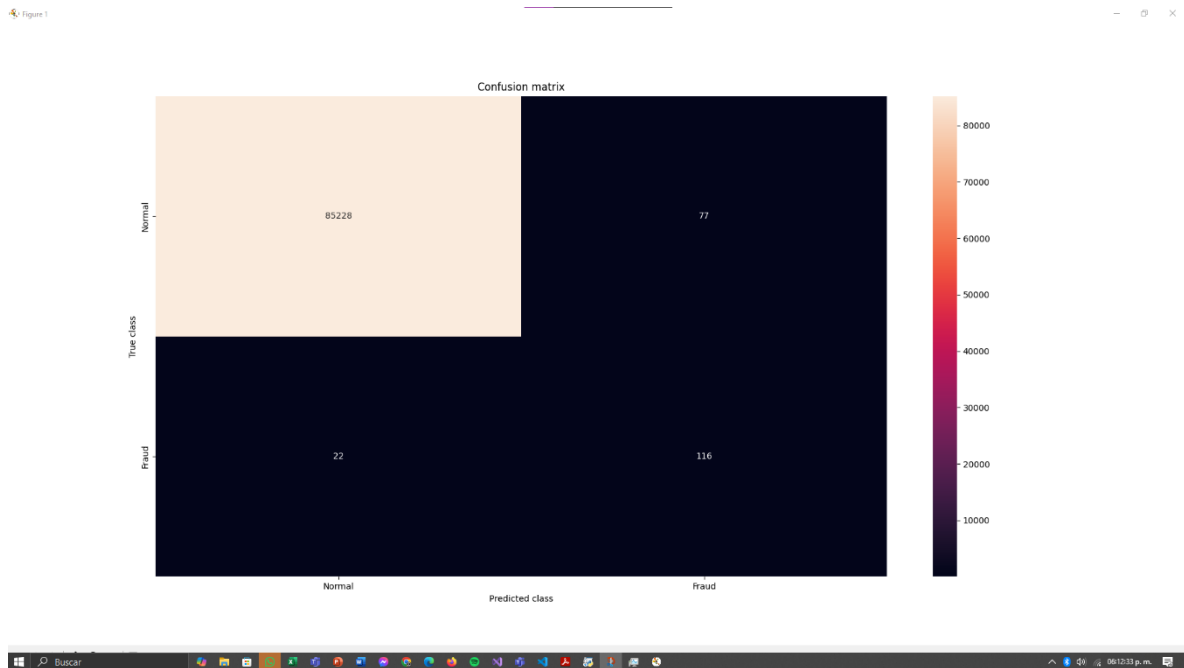


Ilustración 8: Matriz de confusión de Random Forest 2

Resultados Random Forest 2				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	85286
1	0.67	0.83	0.74	157
accuracy			1.00	85443
macro avg	0.84	0.91	0.87	85443
weighted avg	1.00	1.00	1.00	85443

Ilustración 9: Resultados del Modelo calculado por medio de Random Forest 2

3.5 ROC AUC

```
ROC AUC Value:
0.9136433933571733
Presiona Cualquier Tecla para Continuar...
```

Ilustración 10: Calculo del ROC AUC

4 Conclusión

Random Forest es un poderoso algoritmo de aprendizaje automático basado en la combinación de múltiples árboles de decisión para mejorar la precisión y robustez del modelo. Al ser un modelo de ensamble, Random Forest reduce el sobreajuste que podría ocurrir con un solo árbol de decisión, al tiempo que mantiene una alta capacidad de generalización.

La ventaja clave de Random Forest es su capacidad para manejar grandes conjuntos de datos con variables de entrada complejas, adaptarse a distintos tipos de datos (numéricos, categóricos) y ofrecer una interpretación relativamente fácil de los resultados. Además, su capacidad para utilizar múltiples núcleos del procesador para paralelizar el entrenamiento lo convierte en una opción eficiente para tareas de clasificación y regresión.

Aunque suele ser muy efectivo, puede ser más lento en predicciones en comparación con modelos más simples y requiere un ajuste adecuado de hiperparámetros para obtener el mejor rendimiento. A pesar de estas consideraciones, Random Forest sigue siendo una herramienta altamente versátil y popular en el campo del aprendizaje automático.