Act 13: Programando Random Forest en Python

Julio Cesar Torres Marquez

marzo 2025

1 Introducción

Random Forest es un algoritmo de aprendizaje supervisado que utiliza un conjunto de árboles de decisión para hacer predicciones. Cada árbol es entrenado con un subconjunto aleatorio de características, y el modelo final se basa en el voto mayoritario de todos los árboles, lo que ayuda a mejorar la precisión y reducir el sobreajuste.

Este algoritmo es ampliamente utilizado en clasificación debido a su capacidad para manejar grandes conjuntos de datos con una alta dimensionalidad y para proporcionar una medida de la importancia de las características.

2 Metodología

2.1 Descargar el archivo csv de Kaggle

Se descargó el conjunto de datos desde el siguiente enlace de Kaggle:

`https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data`

2.2 Importación de librerías

Se importaron las librerías necesarias para el análisis:

```
import pandas as pd
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
```

2.3 Leer el archivo csv y mostrar los primeros 5 registros

Se cargó el archivo CSV con los datos y se mostraron los primeros registros:

```
dataframe = pd.read_csv(r"creditcard.csv")
print(dataframe.head())
```

2.4 Contar cuántas clases hay

Se contó la distribución de las clases (fraude/no fraude):

```
count_classes = pd.value_counts(dataframe['Class'], sort=True)
```

2.5 Definición de etiquetas y características

Se definieron las etiquetas (y) y las características (X):

```
y = dataframe['Class']
X = dataframe.drop('Class', axis=1)
```

2.6 Definición de conjuntos de entrenamiento y prueba

Se dividieron los datos en conjuntos de entrenamiento y prueba:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7)
```

2.7 Creación y entrenamiento del modelo

Se creó el modelo Random Forest y se entrenó con el conjunto de datos de entrenamiento:

```
model = RandomForestClassifier(n_estimators=100, bootstrap=True, verbose=2, max_features='s
model.fit(X_train, y_train)
```

2.8 Reporte de clasificación

Se generó el reporte de clasificación con las predicciones realizadas en el conjunto de prueba:

```
predictions = model.predict(X_test)
print("Reporte de clasificación")
print(classification_report(y_test, predictions))
```

2.9 Matriz de confusión

Se generó una matriz de confusión para evaluar la precisión del modelo:

```
conf_matrix = confusion_matrix(y_test, predictions)
plt.figure(figsize=(12, 12))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title("Matriz de Confusión")
plt.xlabel("Predicción")
plt.ylabel("Verdadera Clase")
plt.show()
```

3 Resultados

Se obtuvo un buen desempeño en el modelo con la matriz de confusión y el reporte de clasificación que muestra una alta precisión en la predicción de las clases de fraude.

4 Conclusión

El modelo Random Forest ha demostrado ser eficiente para la clasificación de fraude en tarjetas de crédito, superando el riesgo de sobreajuste que podría ocurrir con un solo árbol de decisión. Gracias a su capacidad para manejar grandes volúmenes de datos y características, este modelo es una herramienta poderosa para tareas de clasificación complejas.

5 Referencias Bibliográficas

Bagnato, I. (2020). *Aprende machine learning*. Leanpub.