Reporte de Actividad: Árbol de Decisión con Python

Angel Francisco Hernández Gámez

March 30, 2025

1 Introducción

Un **árbol de decisión** es un algoritmo de aprendizaje automático supervisado utilizado para resolver problemas de clasificación y regresión. Funciona mediante una estructura de decisiones ramificadas, donde los datos son divididos en base a condiciones o características específicas hasta llegar a una decisión final.

Su importancia radica en la facilidad de interpretación, capacidad para modelar relaciones no lineales y utilidad en diversas aplicaciones prácticas.

2 Metodología

Para realizar esta actividad se siguieron los siguientes pasos utilizando el lenguaje de programación Python con la librería Scikit-Learn:

2.1 Carga y Exploración de Datos

Se utilizó un archivo CSV para cargar los datos:

```
import pandas as pd

df = pd.read_csv('datos.csv')
print(df.head())
```

Se realizó un análisis exploratorio inicial verificando:

- Distribución de las variables.
- Existencia de valores faltantes.
- Balance de la clase objetivo.

2.2 Preparación de los Datos

Se realizó un balanceo de datos en caso de ser necesario, utilizando técnicas como undersampling o oversampling:

```
from imblearn.over_sampling import SMOTE

X_balanced, y_balanced = SMOTE().fit_resample(X, y)
```

Los datos categóricos fueron codificados con técnicas como One-Hot Encoding:

```
X_encoded = pd.get_dummies(X_balanced)
```

Los datos fueron divididos en conjuntos de entrenamiento y prueba:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_balanced, test_size=0.2,
```

2.3 Entrenamiento del Modelo

Se entrenó un modelo utilizando un Árbol de Decisión:

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(max_depth=5, random_state=42)
clf.fit(X_train, y_train)
```

3 Resultados

Los resultados fueron evaluados utilizando métricas comunes como la precisión (accuracy), matriz de confusión y un reporte detallado de clasificación:

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n{report}')
print(f'Confusion Matrix:\n{conf_matrix}')
```

3.1 Visualización del Árbol

Se generó una representación visual del árbol:

```
from sklearn.tree import export_graphviz
import graphviz

dot_data = export_graphviz(clf, out_file=None,
                           feature_names=X_encoded.columns,
                           class_names=['Clase0', 'Clase1'],
                           filled=True, rounded=True,
                           special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("arbol_decision")
```

4 Conclusión

El modelo entrenado con el Árbol de Decisión mostró un desempeño satisfactorio en términos generales, destacando:

- Precisión adecuada para la clasificación de las clases evaluadas.
- Facilidad en interpretación de los resultados obtenidos.

Se observó que la selección adecuada de la profundidad del árbol afectó positivamente la capacidad del modelo para generalizar correctamente los datos.

Finalmente, para mejorar aún más los resultados, se podría evaluar el uso de algoritmos como Random Forest o técnicas de ajuste de hiperparámetros más avanzadas.