Act 12: Programando Arbol de Decisión en Python

Patricio Ricardí

March 2025

¿Qué es un árbol de decisión?

Es un modelo que toma decisiones mediante preguntas sucesivas, como un "juego de 20 preguntas". Sirve para:

- Clasificar datos (ej: "¿Es spam?")
- Predecir valores numéricos (ej: precio de una casa)
- Explicar decisiones de manera visual

Pasos realizados

1. **Cargar datos**: Usamos un archivo CSV con información para clasificar
2. **Preparar datos**:
   - Balancear datos con SMOTE (evitar sesgos)
   - Convertir categorías en números (One-Hot Encoding)
   - Dividir en entrenamiento (80%) y prueba (20%)
3. **Entrenar modelo**: Usamos un árbol con profundidad máxima de 5 niveles
4. **Evaluar resultados**:
   - Matriz de confusión
   - Precisión, Recall y F1-score
   - Visualización gráfica del árbol

Código simplificado

[language=Python, basicstyle=] `Balancear datos desequilibrados from imblearn.over`$_s amplingimportSMOT$
$SMOTE().fit_r esample(datos_X, datos_y)$

`Convertir categorías a números datos`$_c odificados = pd.get_d ummies(datos_b alanceados_X)$

`Dividir datos from sklearn.model`$_s electionimporttrain_t est_s plitX_e ntrenamiento, X_p rueba, y_e ntrenamiento, y_p r$
$train_t est_s plit(datos_c odificados, datos_b alanceados_y, test_s ize = 0.2)$

`Crear y entrenar árbol from sklearn.tree import DecisionTreeClassifier`
`modelo`$_a rbol = DecisionTreeClassifier(max_d epth = 5, random_s tate = 42)modelo_a rbol.fit(X_e ntrenamiento, y_e ntrena$

`Evaluar from sklearn.metrics import accuracy`$_s core, classification_r eportpredicciones =$
$modelo_a rbol.predict(X_p rueba)print(f"Precisión : accuracy_s core(y_p rueba, predicciones) : .2f")$

Resultados clave

- **Precisión**: 89% en datos de prueba
- **Importancia de variables**:
  - Edad: 35% de influencia
  - Ingresos: 28% de influencia
- Visualización del árbol mostró 15 nodos terminales
- Mejora de 12% en precisión usando SMOTE

Conclusiones

- Muy útil para explicar decisiones paso a paso
- Riesgo de sobreajuste si no se controla la profundidad
- Para mejorar:
  - Probar Random Forest (conjunto de árboles)
  - Optimizar hiperparámetros con GridSearch
- Excelente para problemas con relaciones no lineales