Práctica 5
Árboles de decisión

Mario Emilio Jiménez Vizcaíno
A01173359@itesm.mx
Tecnológico de Monterrey
Ingeniería en Tecnologías Computacionales
Monterrey, N.L., México

Jesus Abraham Haros Madrid
A01252642@itesm.mx
Tecnológico de Monterrey
Ingeniería en Tecnologías Computacionales
Monterrey, N.L., México

ABSTRACT

En la actualidad, uno de los problemas más importantes para los científicos es la clasificación: el etiquetado de elementos, basándose en las características de estos, seleccionando de un conjunto de clases, una que mejor lo represente. Los árboles de decisión, el objeto de estudio en esta práctica, son clasificadores que predicen las clases de los elementos usando algoritmos simples, lo que facilita su uso e implementación.

1 INTRODUCCIÓN

Un árbol de decisión clasifica instancias de datos planteando una serie de preguntas sobre las características de estos elementos. Cada pregunta se representa con un nodo, y cada nodo apunta a un nodo hijo, que puede ser un nodo terminal (que presenta el resultado del árbol: una clase o etiqueta), u otro nodo de decisión. Las preguntas forman así una jerarquía de decisiones capturada en una estructura de árbol.

Para clasificar un elemento se sigue el camino desde el nodo superior o raíz, hasta un nodo terminal, dependiendo las características del nodo y las preguntas que cada hoja del camino presenten.

Una ventaja de los árboles de decisión es que muchas veces son más interpretables que otros clasificadores, como las redes neuronales y las máquinas de vectores de soporte[1], porque combinan preguntas sencillas sobre los datos de forma comprensible. Por desgracia, pequeños cambios en los datos de entrada pueden provocar a veces grandes cambios en el árbol construido. Los árboles de decisión son lo suficientemente flexibles como para manejar elementos con una mezcla de características de valor real y categóricas, así como elementos con algunas características ausentes.

2 CONCEPTOS PREVIOS

- Programación básica en Python
- Conocimiento de las librerías *scikit-learn*, *matplotlib* y *numpy*
- Conocimientos básicos de estadística

3 METODOLOGÍA

3.1 Dataset Iris

TODO

3.2 Dataset Wine

TODO

3.3 Dataset Breast Cancer

TODO

3.4 Modelo de árbol de decisión

Para la generación del modelo del árbol de decisión se utilizó la implementación de la librería *sklearn*, específicamente la clase *sklearn.tree.DecisionTreeClassifier*[2], que, aunque provee la función de elegir qué algoritmo utilizar para medir la calidad de las preguntas dentro del árbol, utiliza por defecto el algoritmo de impureza de Gini.

El código que ejecutamos para realizar el análisis del dataset se encuentra en el apéndice A.

4 RESULTADOS

TODO

4.1 Dataset Iris

TODO

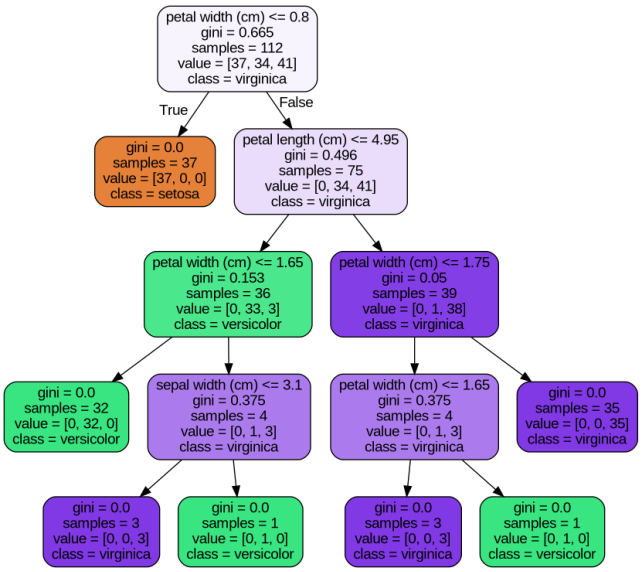

Figure 1: Modelo de árbol de decisión generado para el dataset Iris

4.2 Dataset Wine

TODO

4.3 Dataset Breast Cancer

TODO

5 CONCLUSIONES Y REFLEXIONES

TODO

5.1 Refrexión de Abraham

TODO

5.2 Reflexión de Mario

Por mi parte, considero que esta práctica me ayudó a comprender por qué se utilizan los árboles de decisión a pesar de que toman decisiones ”codiciosas” y con son muy inestables cuando se seleccionan los datos utilizados para entrenar el árbol. Las características que yo pienso hacen del árbol de decisión un clasificador fácil de aprender y enseñar son que se puede representar gráficamente como lo hicimos en la sección resultados, además de que el proceso de clasificación de una instancia o elemento es una serie de preguntas simples.

REFERENCES

[1] Carl Kingsford and Steven L Salzberg. 2008. What are decision trees? *Nature biotechnology* 26, 9 (2008), 1011–1013.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

A CÓDIGO PARA LA GENERACIÓN DEL ÁRBOL DE DECISIÓN

```python
import os

from graphviz import Source
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz

mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# Setup folders
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "decision_trees"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)
os.makedirs(IMAGES_PATH, exist_ok=True)


def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)


# Load and split Iris dataset
iris = load_iris()
xTrain, xTest, yTrain, yTest = train_test_split(
    iris.data, iris.target, random_state=0)

# Create and train the decision tree model
tree_clf = DecisionTreeClassifier(random_state=0)
tree_clf.fit(xTrain, yTrain)

# Export an image of the tree
dot_src = export_graphviz(tree_clf, feature_names=iris.feature_names,
                          class_names=iris.target_names, rounded=True,
                          filled=True)
image_filename = os.path.join(IMAGES_PATH, "iris_tree")
Source(dot_src).render(image_filename, format="png", cleanup=True)

# Test the decision tree
accuracy = tree_clf.score(xTest, yTest)
print("Accuracy:", accuracy)
```