Tecnológico de Monterrey

Nombre: **Roel De la Rosa** **Análisis y Reporte sobre el desempeño del modelo**

Clase: **Inteligencia artificial avanzada para la ciencia de datos**

Profesor: **César Javier Guerra Páramo** Fecha: **18 de Septiembre de 2022**

1 Modelo a usar

Para este avance se ha decidido utilizar el modelo de árboles de decisión. En este tipo de modelos, el algoritmo va generando diferentes criterios, los cuales tras una buena cantidad de observaciones, son capaces de identificar y clasificar a partir de decisiones simples.

2 Set de datos

En este avance se ha decidido utilizar el conjunto de datos de 'Iris'. Este conjunto de datos es bastante conocido y utilizado en el mundo del machine learning. En este conjunto de datos se tienen tres diferentes tipos de flores, el Iris-setosa, Iris-versicolor e Iris-virginica. El objetivo del modelo es que sea capaz de clasificar estas tres tipos de flores a partir de la longitud y el ancho de sus pétalos y sépalos.

3 Análisis

3.1 Separación y Evaluación del modelo

Primero se sepan los datos entre variables dependientes e independientes. En este caso las variables independientes o 'X' son 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',

'PetalWidthCm' y la variable dependiente es 'Species' o 'Y'. Después de esto transformamos estos datos en formato numpy para que puedas ser procesados por el modelo.

Después de esto se utiliza el método train_test_split de Sklearn para separar nuestro set de datos en dos sets completos, uno de entrenamiento y otro de prueba. Después de haber utilizado la función anterior se deberian tener 4 variables, 'X_train', 'Y_train', 'X_test' y 'Y_test'. Siendo dos de ellas usadas para entrenar el modelo y las otras dos para evaluarlo.

Tras haber entrenado el modelo y comparado entre las respuestas predecidas y las reales obtenemos los siguientes resultados:

La precisión del modelo es: 0.9833

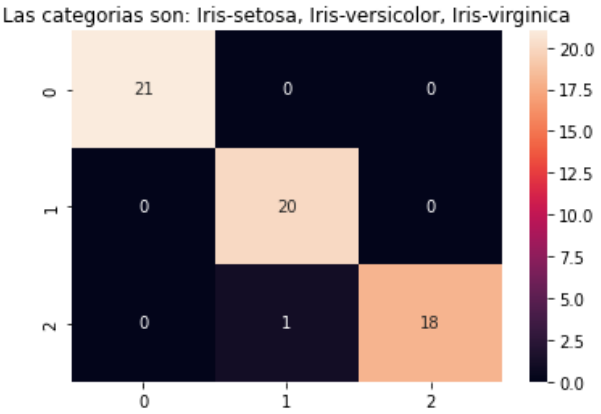

Figure 1: Matriz de confusión de los resultados obtenidos por el modelo y el set de evaluación

3.2 Diagnóstico y explicación del grado de bias y varianza

Por el tipo de modelo que es el de árboles de decisión, se tiene bajo bias pero una varianza bastante alta, pues un pequeño cambio a los datos puede tener un impacto bastante grande al momento de volver a evaluar el modelo.

3.3 Diagnóstico y explicación del nivel de ajuste del modelo

Como se puebe observar en la matriz de confusión de los resultados obtenidos de nuestro modelo, se tienen muy buenos resultados, incluso sin haber ajustado los hiperparámetros.

Se puede observar que se tiene una precisión del 98% y solo se equivoco al clasificar a uno de los 60 casos de prueba.

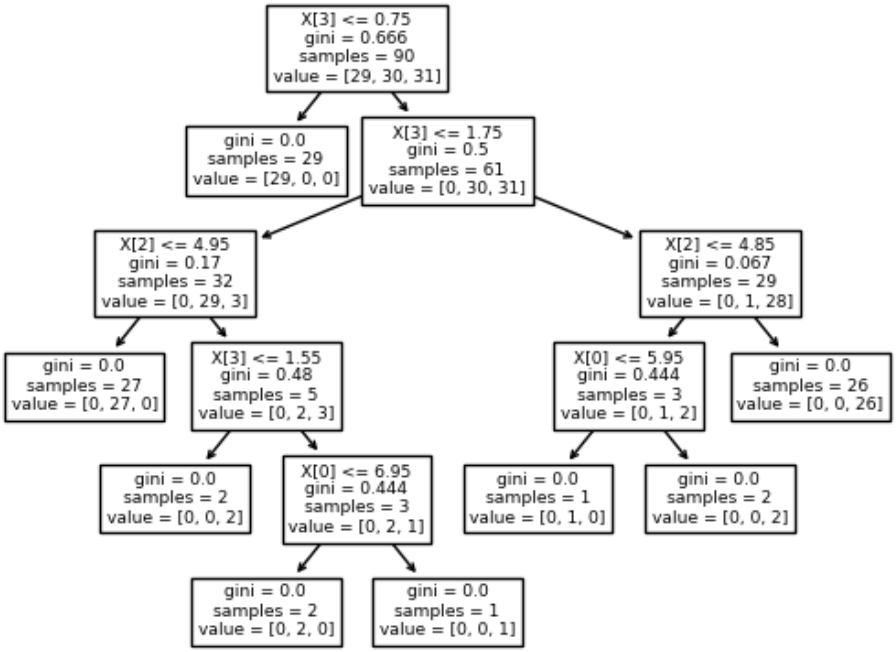

Figure 2: Árbol de decisión

A su vez, también podemos observar en la gráfica anterior como es que el modelo que hicimos toma las decisiones y realiza la clasificación.

4 Técnicas de regularización y ajuste de parámetros

El modelo, al ser un arbol de decisión, tiene algunos hiperparámetros base gracias al framework de Sklearn. En este caso algunos de los más relevantes son "max_depth", el cual se encarga de definir la máxima profundidad que pueden tener las ramas dentro del árbol, "max_leaf_nodes", el cual define la cantidad de ramas que salen por nodos y "criterion", el cual es el criterio con el que se realizan las separaciones de las ramas. En este caso se han utilizado los defaults para todos los hiperparámetros, siendo 'None' en "max_depth" y "max_leaf_nodes" y 'Gini' en "Criterion".

5 Apéndice

```
# Roel Adrian De la Rosa Castillo
# A01197595
# Uso de framework o biblioteca de aprendizaje maquina para la
implementacion de una solucion. - Entrega intermedia

# Importamos las librerias necesarias
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Cargamos nuestro set de datos
data = pd.read_csv('Iris.csv')

# Transformamos los datos para que puedan ser procesados
data = data.drop(['Id'], axis = 1)
X = data.drop(['Species'], axis = 1).to_numpy()
Y = data['Species'].to_numpy()

# Separamos entre set de entrenamiento y prueba
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.4, random_state=26)

# Hacemos el modelo
model = tree.DecisionTreeClassifier()

# Se entrena el modelo con nuestros datos de entrenamiento
```

```
model.fit(X_train, Y_train)

# Se grafica como es que el modelo toma sus decisiones
tree.plot_tree(model)
plt.show()

# Se predice a partir del set de prueba
y_pred = model.predict(X_test)

# Se evalua el modelo y se grafica una matriz de confusion de los
datos que han sido predecidos por el modelo
print("La precision de nuestro modelo es: ",
accuracy_score(Y_test, y_pred))
sns.heatmap(confusion_matrix(Y_test, y_pred), annot=True)
plt.title('Las categorias son: Iris-setosa, Iris-versicolor,
    Iris-virginica')
plt.show()
```

5.1 Referencias

- Bhalla, D. (n.d.). Understanding bias-variance tradeoff. ListenData. Retrieved September 11, 2022, from https://www.listendata.com/2017/02/bias-variance-tradeoff.html#:~:text=An%20algorithm%20like%20Decision%20Tree,It%20leads%20to%20overfitting.&text=It%20means%20there%20is%20a,of%20pattern%20outside%20training%20data.

- Brownlee, J. (2019, October 24). Gentle introduction to the bias-variance trade-off in machine learning. Machine Learning Mastery. Retrieved September 11, 2022, from https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/