Reporte de Laboratorio Nro. 1

Jorge Pineda[L00385745]

Universidad de las Fuerzas Armadas
japineda@espe.edu.ec

Tema: Arboles de decisión

Resumen

El algoritmo de árboles de toma de decisiones es uno de los más implementados, ya que, este se encuentra relacionado con tres características, las cuales son, la interpretabilidad, la eficiencia y la flexibilidad. Por el hecho de que el árbol de decisión puede ser implementado en diversos problemas, como de clasificación y de regresión, por lo cual el aprendizaje automático de un árbol de decisión se caracterizara porque implementa la lógica y las matemáticas con el fin de generar diversas reglas en lugar de seleccionar, basándose solamente en la intuición y la subjetividad. En el cual se implementaran los criterios y los algoritmos poda, los cuales nos ayudaran a controlar la complejidad y optimizar el rendimiento del árbol de decisión.

1. Introducción

El modelo de árboles de decisión es el más conocido y desarrollado entre los métodos de aprendizaje autónomo. Ya que este se utiliza a menudo en minería de base de datos, cuyo fin es realizar tareas de predicción y de diagnóstico. Además, se implementan problemas de clasificación, regresión o de predicción dependiente del tiempo. El punto fuerte de los árboles de toma de decisiones son sus características de interpretabilidad, puesto que es un método gráfico, el cual se encuentra diseñado para problemas que implicaran una secuencia de decisiones y eventos sucesivos [1].

De una manera más concreta, los resultados del árbol de toma de decisiones van a formalizar el razonamiento con el fin de poder reproducir una secuencia de decisiones y encontrar una característica del un objeto. Una de sus principales ventajas es que este tipo de modelo el uno de los más fáciles de comprender y los puede producir con fácilmente la secuencia de decisiones para así poder predecir la categoría del objeto. Por lo cual los resultados que proporcionan son de una estructura gráfica o será una base de reglas que facilitara la compresión, además corresponderá con el razonamiento humano. Puesto que el aprendizaje por árbol de toma de decisiones va a formar parte del aprendizaje supervisado, en el cual se da la clase de cada objeto de la base de datos [1]

Una vez se haya construido el modelo tendremos la capacidad de poder extraer un conjunto de reglas de clasificación, en este tipo de modelo las reglas extraídas se podrán utilizar para clasificar nuevos objetos en la cual cuya clase se desconozca. La clasificación es realizada recorriendo un camino el cual va desde la raíz hasta una hoja. En cambio, la clase devuelta o clase por defecto es una de las más frecuente ente los ejemplos de hoja. Puesto que en cada nodo interno del árbol existirá una pregunta, la cual va a corresponder a un atributo de la base de aprendizaje y una

rama va a corresponder a cada uno de los posibles valores del atributo. En cada nodo hoja habrá un valor de clase, es decir, un camino desde la raíz hasta el nodo correspondiente, por ello una serie de atributos con sus valores, serán la estructura en forma de diagrama de flujo con una partición recursiva que ayudara al usuario en la toma de decisiones [1]

2. Método

En este laboratorio se realizo la implementación del algoritmo de árbol de decision con el dataset Diabetes, primero se empezó con la importacion de librerias necesarias para el laboratorio como se muestra en el siguiente codigo.

```
'''Aqui se a adio las librerias necesarias que se usaran mas adelante '''
import pandas as pd
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix
from six import StringIO
from IPython.display import Image
import pydotplus
import matplotlib.pyplot as plt
import seaborn as sns
```

Listing 1: Librerías usadas para este laboratorio

Una vez importada todas las librerias requeridas, comenzamos con la carga del dataset diabetes

```
'''Ahora cargaremos el dataset diabetes '''
dfDiabetes = pd.read_csv("Dataset of Diabetes.csv")
'''Mostraremos el dataset que usaremos en el laboratorio'''
dfDiabetes.head()
```

Listing 2: Librerías usadas para este laboratorio

Con el siguiente codigo lo realizamos para ver las columnas que existen en el dataset y el tipo de dato que poseen.

```
'''Veremos todas las columnas que tiene nuestro dataset'''
dfDiabetes.keys()

'''Veremos los datos de la columna CLASS'''
dfDiabetes.groupby("CLASS").size()

'''Tambien veremos los datos de la columna Gender'''
dfDiabetes.groupby("Gender").size()
```

Listing 3: Librerías usadas para este laboratorio

Como observamos que los datos no estaban limpios, procedimos a modificarlos y a visualizarlos con el siguiente código.

```
'''Como observamos que estos datos no estan correctamente para ser tratados,
    realizaremos unos cambios para poder trabajarlos'''
'''Modificar la mayuscula'''
dfDiabetes['Gender'] = dfDiabetes['Gender'].replace('f','F')
'''Modificar el espaciado'''
```

```
dfDiabetes['CLASS'] = dfDiabetes['CLASS'].replace('N ','N')
dfDiabetes['CLASS'] = dfDiabetes['CLASS'].replace('Y ','Y')

'''Comprobacion de la columna CLASS'''
dfDiabetes.groupby("CLASS").size()

'''Comprobacion de la columna Gender'''
dfDiabetes.groupby("Gender").size()
```

Listing 4: Librerías usadas para este laboratorio

Una vez comprobado que ya se trato los datos del data set, se procedio a realizar un diccionario de datos para convertir los datos en numeros para poder ser procesados con ayuda del mapeo de datos.

```
'''Realizaremos un diccionario de datos para poder convertirlos en numeros'''
diccGender = {'M': 0, 'F': 1}
diccClass = {'N': 0,'Y': 1, 'P': 2}
'''Aplicaremos los diccionarios creados mediante el mapeo de cada columna'''
dfDiabetes['Gender'] = dfDiabetes['Gender'].map(diccGender)
dfDiabetes['CLASS'] = dfDiabetes['CLASS'].map(diccClass).fillna(0.0).astype(int)
'''Veremos los cambios afectados, ahora todo son numeros'''
dfDiabetes
```

Listing 5: Librerías usadas para este laboratorio

Una vez convertido el tipo de dato mediante el mapeo de nuestro dataset corroboramos que todo este correcto con la siguiente linea de codigo

```
'''Comprobaremos el tipo de dato que tiene nuestro dataset, para poder ser
    trabajado'''
dfDiabetes.info()
```

Listing 6: Librerías usadas para este laboratorio

Ahora continuaremos con la seleccion de variables para poder entrenar el modelo, asignando las variables corespondiende del dataset.

```
'''Seleccionaremos las caracteristicas para el modelo'''
features = [ 'Gender', 'AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG', 'HDL', 'LDL','
    VLDL', 'BMI' ]
'''Asignaremos las caracteristicas a la variable X'''
X = dfDiabetes[features]
'''Comprobacion de la variable X'''
X

'''Asignamos la etiqueta a la variable y'''
y = dfDiabetes['CLASS']
'''Comprobacion de la variable y'''
y
```

Listing 7: Librerías usadas para este laboratorio

Una vez definidas las variables, procedemos a entrenar el modelo de árbol de decisión, definiendo los parámetros necesarios.

```
'''Clasificador para el arbol de decision'''
clf = DecisionTreeClassifier()

'''Establecemos los par metros para el modelo, con un 70% en entrenamiento y un
    30% en test '''
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
    random_state=1)

'''Entrenando al modelo'''
clf = clf.fit(X_train, y_train)

'''Almacenando las predicciones en una variable'''
y_pred = clf.predict(X_test)
```

Listing 8: Librerías usadas para este laboratorio

En esta sección obtendremos las métricas del modelo, obteniendo la precisión del algoritmo realizado.

```
'''Calculamos la precision del algoritmo'''
Precision = format(accuracy_score(y_test, y_pred))
print('Precision:'+ Precision)
'''Corroboramos mediante la tabla de reporte de clasificacion del algoritmo'''
print(classification_report(y_test, y_pred, target_names=['CLASS N', 'CLASS P', '
    CLASS Y']))
```

Listing 9: Librerías usadas para este laboratorio

Una ves realizado el modelo del algoritmo de arbol de decision, procedemos a graficarlas siguientes lineas de comandos.

```
'''
Para poder graficar el arbol de decision tendremos que importar la funci n
    export_graphviz y un par de clases
adicionales de sklearn, IPython y pydotplus
'''
''' Primero se creara una variable para almacenar la cadena'''
dot_data = StringIO()
'''Ahora para poder visualizarlo como imagen tendremos que usar la libreria para
    convertirlo en png'''
export_graphviz(clf, out_file = dot_data,filled = True, rounded = True,
    special_characters = True,feature_names = features,class_names = ['0', '1','2'
    ])
'''Ahora le damos al grafico los datos de la variable creada al inicio'''
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
'''Ahora se almacenara la imagen en un png'''
graph.write_png('DiabetesArbol.png')
'''Mostrar imagen'''
Image(graph.create_png())
```

Listing 10: Librerías usadas para este laboratorio

Como vimos el arbol salio muy extenso, asique se realizo una poda del arbol, disminuyendo su tamaño peor no afectando a la importancia de la informacion.

```
'''
Como el arbol es muy extenso para visualizarlo y entenderlo, lo recortaremos para
    que tenga un mejor tama o
sin afectar su informacion, con una profundidad de 4
'''
clf = DecisionTreeClassifier(criterion="entropy",max_depth=3)
'''Almacenamos el entrenamiento'''
clf = clf.fit(X_train, y_train)
'''Almacenamos la prediccion'''
y_pred = clf.predict(X_test)
'''Creamos variable para la cadena'''
dot_data = StringIO()
```

```
'''Ahora para poder visualizarlo como imagen tendremos que usar la libreria para convertirlo en png'''
export_graphviz(clf, out_file = dot_data, filled = True, rounded = True, special_characters = True, feature_names = features, class_names = ['0', '1','2'])
'''Ahora le damos al grafico los datos de la variable creada al inicio'''
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
'''Ahora se almacenara la imagen en un png'''
graph.write_png('DiabetesArbolPodado.png')
'''Mostrar imagen'''
Image(graph.create_png())
```

Listing 11: Librerías usadas para este laboratorio

Ahora procederemos a realizar la matriz de confusión de nuestro modelo.

```
'''Ahora crearemos la matriz de confusion con ayuda de y_test y y_pred'''
MatrizConfusion = confusion_matrix(y_test, y_pred)
'''Visualizaremos la matriz de confusion'''
print(MatrizConfusion)
```

Listing 12: Librerías usadas para este laboratorio

Una vez realizada la matriz de confusion, lo graficaremos en formato de imagen para poder guardarlo.

```
'''Ahora para poder visualizar la matriz en formato figura usaremos heatmap'''
sns.heatmap(MatrizConfusion,annot=True,fmt="d")
'''Creamos un encabezado al grafico'''
plt.title("Matriz de Confusi n", position=(0.5, 0.9))
'''Ponemos un titulo al eje X '''
plt.xlabel('prediccion');
'''Ponemos un titulo al eje y'''
plt.ylabel('actual');
'''Guardamos la figura en una imagen png'''
plt.savefig('matrizConfusion.png')
```

Listing 13: Librerías usadas para este laboratorio

3. Results and Analysis

Como resultados de la guia del laboratorio 1 pudimos obtener el árbol de decisión referente al dataset Diabetes donde los datos ya fueron procesados y logramos obtener el siguiente resultado a la hora de graficarlo como se muestra en la (Figura 1).

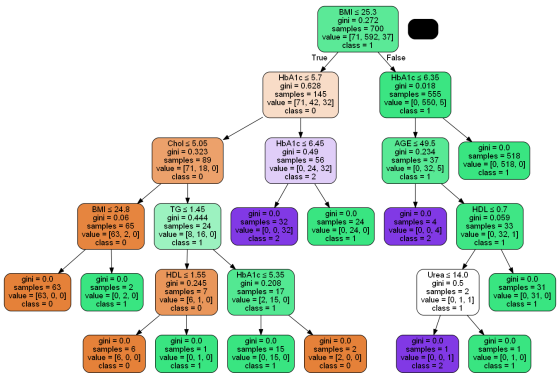

Figura 1: Árbol Diabetes.

Como el árbol era muy frondoso, procedimos a podarlo, de tal manera que ajustamos las dimensiones del árbol para un mejor entendimiento de la imagen, dándonos el siguiente resultado como se muestra en la (Figura 2).

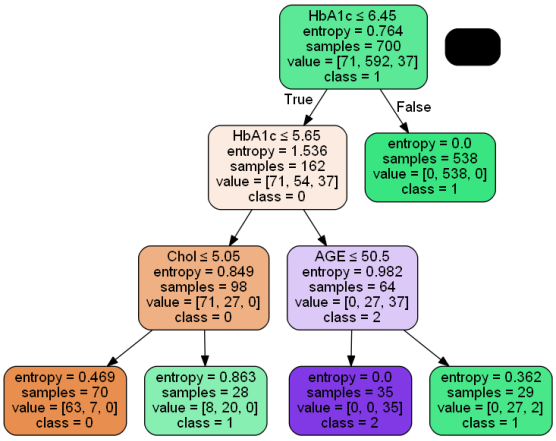

Figura 2: Árbol Diabetes Podado.

También se logro realizar la matriz de confusión con los resultados obtenidos, dándonos una matriz tri dimensional como se muestra en la Figura 3).

A continuación, adjunto el enlace al repositorio github donde se encuentra todo el versiona miento realizado para este laboratorio

Repositorio del laboratorio 1

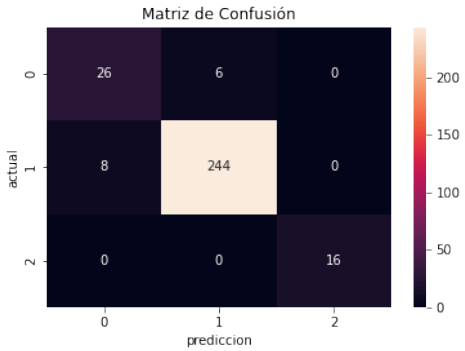

Figura 3: Matriz de Confusión.

4. Discusión

¿Qué es un modelo de árboles de decisión y cuál es su utilidad?

El árbol de decisión es un modelo predictivo el cual es una especie de mapa en el que se muestra cada una de las opciones de decisión posibles y sus resultados el cual es muy útil ya que permite compara diferentes decisiones y acciones según las necesidades, probabilidades y beneficios. Las ventajas y diferencias de los modelos de árbol de decisión son las siguientes, como una de sus principales ventajas tenemos que es simple de entender e interpretar, además su diagrama es sencillo por lo cual se puede visualizar fácilmente lo cual facilitara el entendimiento del proceso, no requiere la colocación de datos complejos, se puede implementar tanto en variables cuantitativas como en cualitativas, los árboles de decisión son maleables ya que se puede agregar nuevas opciones, además que se puede combinar de manera fácil con diversas herramientas de toma de decisiones. Como desventajas tenemos que son inestables ya que al realizar un pequeño cambio podemos suponer un árbol completamente diferente, además de no poder garantizar que el árbol que se ha generado sea el mas óptimo, y si no se posee mucho conocimiento sobre la creación de estos arboles estos pueden quedar sesgados, lo cual llevara a que el árbol pueda llegar a ser muy complejo perdiendo así su utilidad.

5. Conclusión

Mediante la implementación de la base de datos diabetes se consiguió tratar los datos con el fin de poder procesarlos para aplicar el algoritmo de árbol de decisión, en este proceso se definió los valores de X y para poder entrenar el modelo para que posteriormente se obtuviera sus métricas.Por lo cual se realizo la gráfica del árbol de decisiones y se podo para un mejor entendimiento como también se realizo de manera exitosa la matriz de confusión donde visualizar los resultados de manera matricial.Todas estas figuras se almacenaron y se guardaron en formato png y se realizo los versionamos correspondientes.

Referencias

[1] Bouchra Lamrini. Contribution to decision tree induction with python: A review. *researchgate*, 2020.