Unidad Profesional Interdisciplinaria de Ingeniería campus Tlaxcala

INSTITUTO POLITÉCNICO NACINAL

FUNDAMENTOS DE INTELIGENCIA ARTIFICIAL.

Algoritmo de Árboles de Decisión aplicado a la base de datos de Golf.

Equipo:

Navil Pineda Rugerio.

Luis Daniel Islas García.

Miguel Ángel Sánchez Zanjuampa

15 de junio, 2023

Resumen
Un árbol de decisión es un algoritmo de aprendizaje supervisado no paramétrico que se utiliza tanto para tareas de clasificación como de regresión. Tiene una estructura de árbol jerárquico, que consta de un nodo raíz, ramas, nodos internos y nodos hoja. Los árboles de decisión se utilizan ampliamente en el campo del aprendizaje automático y la inteligencia artificial para tomar decisiones o predecir resultados. Se representan visualmente como un árbol, en el que cada nodo interno representa una prueba sobre un atributo, cada rama el resultado de la prueba y cada hoja una clase o un valor de predicción. En esta práctica se diseñará un algoritmo de árbol de decisión aplicado a una base de datos sobre condiciones adecuadas para jugar un partido de Golf, que permita realizar predicciones evaluando dichas condiciones como adecuadas o no.

Palabras clave: Algoritmo de predicción, árboles de decisión, dataset de Golf, entropía, gini index.

Abstract
A decision tree is a non-parametric supervised learning algorithm used for both classification and regression tasks. It has a hierarchical tree structure, consisting of a root node, branches, internal nodes, and leaf nodes. Decision trees are widely used in the field of machine learning and artificial intelligence to make decisions or predict outcomes. They are represented visually as a tree, where each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf represents a class or a prediction value. In this practice we will design a decision tree algorithm applied to a database on suitable conditions to play a golf game, which allows us to make predictions by evaluating these conditions as suitable or not.

Keywords: Prediction algorithm, decision trees, Golf dataset, entropy, gini index.

I. MARCO TEÓRICO.

“Un árbol de decisión es un algoritmo de aprendizaje supervisado no paramétrico, que se utiliza tanto para tareas de clasificación como de regresión. Tiene una estructura de árbol jerárquica, que consta de un nodo raíz, ramas, nodos internos y nodos hoja” (IBM, s.f).

Los árboles de decisión son una técnica popular en el campo del aprendizaje automático y la inteligencia artificial que se utiliza para tomar decisiones o predecir resultados. Se basan en la representación visual de un árbol, donde cada nodo interno representa una prueba en un atributo, cada rama representa el resultado de la prueba y cada hoja representa una clase o un valor de predicción.

En concreto, el aprendizaje mediante ´arboles de decisión es un método de aproximación de una función objetivo de valores discretos en el cual la función objetivo es representada mediante un árbol de decisión.

Los árboles de decisión se utilizan en una amplia gama de aplicaciones, como la clasificación de datos, la toma de decisiones en tiempo real y el descubrimiento de conocimiento.

A partir de un árbol de decisión se permite la clasificación de nuevos casos siempre y cuando no existan modificaciones sustanciales en las condiciones bajo las cuales se generaron los ejemplos que sirvieron para su construcción. Además, este tipo de algoritmo reduce el número de variables independientes.

Un árbol de decisión consta de varios elementos fundamentales:

1. **Nodo raíz:** Es el nodo principal del árbol y representa la variable o atributo más relevante para la clasificación o predicción. A partir de este nodo, se ramifican los nodos subsiguientes.
2. **Nodos internos**: Son los nodos intermedios entre la raíz y las hojas. Cada nodo interno representa una prueba o condición sobre un atributo y tiene ramas que conducen a otros nodos o hojas.
3. **Hojas:** Son los nodos terminales del árbol y representan las clases o los valores de predicción. Cada hoja tiene asociada una clase o un valor determinado.
4. **Ramas**: Son los enlaces que conectan los nodos y representan el resultado de una prueba en un atributo. Cada rama está asociada con un valor específico del atributo.
5. **Atributos:** Son las características o variables que se utilizan para realizar las pruebas en los nodos internos. Los atributos pueden ser numéricos o categóricos.

La construcción de un árbol de decisión implica varias etapas:

1. Selección del atributo de mayor relevancia: Se utiliza una medida de impureza (como la ganancia de información o el índice Gini) para determinar el atributo que mejor separa los datos y proporciona una mayor ganancia de información o una menor impureza.
2. División de los datos: Se realiza una partición de los datos de entrenamiento en base al atributo seleccionado en el paso anterior. Cada rama del árbol representa uno de los posibles valores del atributo.
3. Recursividad: El proceso se repite de manera recursiva en cada subconjunto de datos resultante de la división, construyendo así el árbol de decisión completo, primero recorriendo los nodos del lado izquierdo y posteriormente los nodos del lado derecho.

Objetivo

El objetivo de esta práctica es diseñar un algoritmo de árboles de decisión aplicado a una base de datos sobre las condiciones adecuadas para jugar un partido de Golf, que permita hacer predicciones evaluando esas condiciones ya sean aptas o no. Se diseñará un programa en Python que realice la lectura del archivo de la base de datos, así como la construcción del árbol de decisión desde cero, sin utilizar ninguna librería para aprendizaje automático.

II. Materiales

- Bibliotecas y herramientas para leer los datos de un archivo, para el manejo de arreglos y la visualización de los datos.
- Funciones y clases para la lectura del archivo y el algoritmo de árboles de decisión.
- Jupyter Notebook.

III. Desarrollo

A. *Librerías*

Se importan las librerías, utilizamos numpy para las operaciones entre los vectores de datos, pandas para mostrar la información, Counter de collections para la función que construye el árbol de decisión, y la librería anytree para imprimir el árbol de decisión.

```
import numpy as np
import pandas as pd
from collections import Counter
from anytree import Node, RenderTree
```

B. *Cargar el DataSet.*

Para cargar la base de datos de Golf se utiliza la función de pandas read_csv y se pasa el archivo que contiene todos los atributos y sus respectivas conclusiones acerca de si es posible jugar el partido de Golf o no.

Cargar DataSet

```
df = pd.read_csv('golf_data.csv')
```

C. *Modificar atributos por palabras completas.*

Lo siguiente tiene el propósito de visualizar mejor los datos, para que el resultado de una futura predicción sea más fácil de comprender.

Para ello accedemos al Data Frame en el que fue leído el archivo, y en cada columna de la tabla modificamos los valores por palabras completas, en donde corresponda.

Modificar los atributos de las caracteristicas por palabras completas

```
df["Ambiente"] = df["Ambiente"].map({"S": "Soleado", "N": "Nublado", "Ll": "Lluvioso"})
df["Temperatura"] = df["Temperatura"].map({"A": "Alta", "M": "Media", "B": "Baja"})
df["Humedad"] = df["Humedad"].map({"A": "Alta", "N": "Normal"})
df["Viento"] = df["Viento"].map({"S": "Si", "N": "No"})
df["Jugar"] = df["Jugar"].map({"S": "Si", "N": "No"})
```

D. *Visualización de los datos modificados.*

A continuación, se muestra el diccionario de datos para comprobar que los valores han sido modificados correctamente.

```
df
```

	Ambiente	Temperatura	Humedad	Viento	Jugar
0	Soleado	Alta	Alta	No	No
1	Soleado	Alta	Alta	Si	No
2	Nublado	Alta	Alta	No	Si
3	Lluvioso	Media	Alta	No	Si
4	Lluvioso	Baja	Normal	No	Si
5	Lluvioso	Baja	Normal	Si	No
6	Nublado	Baja	Normal	Si	Si
7	Soleado	Media	Alta	No	No
8	Soleado	Baja	Normal	No	Si
9	Lluvioso	Media	Normal	No	Si
10	Soleado	Media	Normal	Si	Si
11	Nublado	Media	Alta	Si	Si
12	Nublado	Alta	Normal	No	Si
13	Lluvioso	Media	Alta	Si	No

E. Algoritmo de Árbol de Decisión.

Para construir un árbol de decisión utilizando los datos del DataSet creamos una clase llamada DecisionTree. Esta clase tiene un constructor que no recibe ningún parámetro, pero inicializa como None al atributo self.tree.

F. Entropía.

Cómo ya conocemos, un árbol de decisión utiliza la entropía para buscar el nodo raíz, la entropía calcula la cantidad de desorden de la información, y depende de la cantidad de datos en un nodo.

Para ello se calcula la proporción o ratio de una categoría:

$$Entropia(S) = \sum_{i=1}^{n} -p_i \cdot log_2(p_i)$$

En el programa se utiliza la función entropy, la cual recibe como parámetro la data que ya se recuperó del archivo. En esta función se obtienen los valores únicos que hay en los datos para obtener la entropía de cada uno, también se obtiene el total de datos del archivo. Después se itera sobre cada uno de los valores únicos de los atributos y se calcula la proporción de ese atributo, esto es la probabilidad de la clase, dividiendo el numero de atributos que tienen el valor iesimo sobre el total de atributos. Después se calcula la entropía siguiendo la fórmula anterior, y se devuelve ese valor.

```
# Funcion pata encontrar la entropia de cada atribut
def entropy(self, data):
    # Obtener los atributos
    counter = Counter(data)
    # Contar el numero de atributos totales
    total = len(data)
    # Inicializar entropia a 0
    entropy = 0

    # Iterar cada valor de los atributos para obtene
    for value in counter.values():
        # Obtener la proporcion de ese atributo (prob
        proportion = value / total
        # Calcular la entropia como la resta de la pr
        entropy -= proportion * np.log2(proportion)

    # Regresar la entropia total
    return entropy
```

G. Gini Index

Así como se tiene una función para determinar el nivel de desorden que hay en un nodo, también se tiene una función para determinar la pureza del nodo, para ello se utiliza la función information_gain la cual recupera los valores de los atributos y con ello se calcula la entropía total con la función anterior, pasando la data del atributo “Jugar”, que es el que se quiere predecir.

Después se calcula el indice de gini para cada valor único en los atributos, utilizando la siguiente fórmula.

$$GINI(t) = 1 - \sum_{i=1}^{n} (P_i)^2$$

H. Función para obtener el atributo con mayor ganancia.

La siguiente función obtiene la ganancia de cada atributo y mediante la función max se obtiene el nodo que tiene el mayor valor. Este será un nodo puro.

```
# Funcion para elegir el nodo que tiene la mayor ganancia de in
def get_best_attribute(self, data, attributes):
    gains = {attribute: self.information_gain(data, attribute)
             for attribute in attributes}
    return max(gains, key=gains.get)
```

I. Función para crear el árbol de decisión.

Esta función utiliza recursión para buscar el nodo raíz y los nodos más puros para todos los atributos, primero carga todas las clases a predecir en una variable y.

Se tienen dos condiciones de paro:

1. Si solo hay un atributo regresa ese valor, pues ya no hay que buscar más.
2. Si ya no hay más atributos a la derecha o a la izquierda de la búsqueda regresa la clase que predomina en ese último nodo.

Si ninguna de esas condiciones se cumple, el algoritmo continúa con dividir la información, se obtiene el nodo raíz de ese árbol o subárbol y se calculan los demás nodos raíz, u hoja para los demás subárboles.

Finalmente cuando se tiene todo el árbol construido se guarda en la variable self.tree y se retorna.

```
def create_decision_tree(self, data, attributes, target_at
    # Inicializar las clases a predecir
    y = data[target_attribute]

    # Si solo hay una clase regresar ese valor
    if len(set(y)) == 1:
        return y.iloc[0]

    # Si ya no hay mas atributos a la derecha o a la izqui
    if len(attributes) == 0:
        major_value = Counter(y).most_common(1)[0][0]
        return major_value
```

```
# Sino obtener el mejor nodo para dividir la informacion
best_attribute = self.get_best_attribute(data, attributes)
tree = {best_attribute: {}}
attribute_values = set(data[best_attribute])

# Encontrar los demas nodos, pasando recursivamente los subarboles
for value in attribute_values:
    subset = data[data[best_attribute] == value].drop(best_attribute, axis=1)
    subtree = self.create_decision_tree(subset, attributes -
                                        {best_attribute}, target_attribute)
    tree[best_attribute][value] = subtree

# Asignar al arbol de la clase el nuevo arbol encontrado
self.tree = tree
return self.tree
```

Finalmente, se tiene una función para imprimir el árbol el cual utiliza la librería anytree, que recibe un árbol en forma de diccionario de datos, y se imprime gráficamente.

```
# Funcion para imprimir el arbol que se encuentra
def build_tree(self, data, parent=None):
    for key, value in data.items():
        node = Node(key, parent=parent)
        if isinstance(value, dict):
            self.build_tree(value, parent=node)
        else:
            Node(value, parent=node)
```

IV. Resultados

Ahora se puede utilizar el algoritmo para hacer predicciones con los datos del tiempo actual, por ejemplo, esto se logra visualizando los datos.

Lo primero que se realiza es entrenar el modelo, la data que pasamos es el Data Frame anteriormente creado, y los atributos que se convertirán en nodos se obtienen quitando la columna “Jugar” (que corresponde a las clases).

```
data = df
attributes = set(data.keys()) - {'Jugar'}
attributes
```

```
{'Ambiente', 'Humedad', 'Temperatura', 'Viento'}
```

Se inicializa el árbol utilizando la clase DecisionTree(), y se guarda en el objeto model.

Inicializar el arbol de decision

```
model = DecisionTree()
```

Después, se crea el árbol utilizando la clase create_decision_tree, y se pasa la data, los atributos, y las clases a predecir.

Crear el arbol de decision con la clase create_decision_tree

```
: decision_tree = model.create_decision_tree(data, attributes, target_attribute='Jugar'
  decision_tree
```

```
: {'Ambiente': {'Soleado': {'Humedad': {'Normal': 'Si', 'Alta': 'No'}},
    'Nublado': 'Si',
    'Lluvioso': {'Viento': {'Si': 'No', 'No': 'Si'}}}}
```

Como se puede observar al imprimir la variable decisión tree ya se tiene el árbol en forma de diccionario de datos.

Por último, se visualiza el árbol ya como tal como una estructura, utilizando la función build_tree.

```
root = Node("")
model.build_tree(decision_tree, parent=root)

for pre, fill, node in RenderTree(root):
    print(f"{pre}{node.name}")
```

```
└── Ambiente
    ├── Soleado
    │   └── Humedad
    │       ├── Normal
    │       │   └── Si
    │       └── Alta
    │           └── No
    ├── Nublado
    │   └── Si
    └── Lluvioso
        └── Viento
            ├── Si
            │   └── No
            └── No
                └── Si
```

A partir de los siguiente ya podemos decir que, si el ambiente está soleado, y el nivel de humedad es normal, es posible jugar un partido de Golf. O bien que si el ambiente está nublado no hay problema, también se puede jugar un partido de Golf.

V. Conclusiones

En esta práctica, se ha aplicado el algoritmo de árboles de decisión al conjunto de datos sobre las condiciones adecuadas para jugar un partido de golf. El objetivo fue construir un programa en Python desde cero, sin utilizar ninguna librería de aprendizaje automático, para generar un árbol de decisión que pudiera realizar predicciones sobre si las condiciones eran aptas o no para jugar. El uso de árboles de decisión resultó ser una elección adecuada para este problema, ya que permitió representar de manera intuitiva y comprensible las reglas de decisión basadas en las diferentes condiciones.

Finalmente, el árbol de decisión generado a partir del conjunto de datos del golf demostró ser capaz de hacer predicciones sobre las condiciones adecuadas para jugar un partido. Al evaluar nuevas instancias en el árbol, el algoritmo pudo clasificarlas correctamente según las reglas aprendidas durante la construcción del árbol.

Referencias

[1] Hojas, I. M. (2020, enero 21). *Construyendo árboles de decisión*. StatDeveloper; Ignacio Moreno Hojas. https://www.statdeveloper.com/construyendo-arboles-de-decision/

[2] *¿Qué es un árbol de decisión?* (s/f). Ibm.com. Recuperado el 16 de junio de 2023, de https://www.ibm.com/mx-es/topics/decision-trees

[3] Ferrero, R. (s/f). *Qué son los árboles de decisión y para qué sirven*. Máxima Formación. Recuperado el 16 de junio de 2023, de https://www.maximaformacion.es/blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven/