Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2433 - Minería de Datos

Tarea 2: Árbol de decisión

Profesor Vicente Domínguez
26 de Octubre de 2020

Indicaciones

- Fecha de Entrega: 26 de Octubre a las 23:59.
- Se debe entregar la tarea en el repositorio asignado a cada uno por Github Classroom. Deben ir siguiente link https://classroom.github.com/a/Cs7b4fNE.
- Cada hora de atraso descuenta 1 punto de la nota que obtengas.
- La tarea es *individual*. La copia será sancionada con una nota 1.1 en la tarea, además de las sanciones disciplinarias correspondientes.
- **No se permite** el uso de librerías diferentes a las mencionadas en este enunciado.
- Todo código entregado se comparará con implementaciones del algoritmo en internet para evitar la copia de este.

Introducción

Esta tarea consiste en implementar el algoritmo **Decision Tree**. Este es un método de aprendizaje supervisado no paramétrico que se utiliza para clasificación y regresión. El objetivo es crear un modelo que prediga el valor de una variable objetivo aprendiendo reglas de decisión simples inferidas de las características de los datos.

Para esta tarea deberás programar el modelo de Árbol de Decisión con una serie de especificaciones. Para probar el rendimiento del algoritmo se deberá utilizar un conjunto de datos provenientes de proyectos de Kickstarter.

Kickstarter es un sitio web de micromecenazgo para proyectos creativos. Mediante Kickstarter se ha financiado una amplia gama de esfuerzos, que van desde películas independientes, música y cómics a periodismo, videojuegos y proyectos relacionados con la comida. Dada la información de diferentes proyectos, se espera entregar un árbol de decisión capaz de dar el resultado final del proyecto: exitoso o fallido.

Base de Datos

La base de datos a utilizar corresponde a una gran variedad de proyectos lanzados durante el 2016, con diferentes *features* de cada proeycto y con una etiqueta que dice su resultado (exitoso o fallido). Este dataset fue extraído de Kaggle. A continuación se presenta un detalle de las columnas que presenta la base de datos:

- `ID`: identificador único por proyecto.
- `name`: nombre del proyecto.
- `category`: categoría específica a la cual pertenece el proyecto. Ejemplo de categorías sería: *Conceptual Art*, *Digital Art* o *Illustration*.
- `main_category`: macro categoría a la cual pertenece el proyecto. Por ejemplo, las 3 categorías mencionadas anteriormente estarían dentro de la macro categoría llamada *Art*.
- `goal`: cantidad de dolares que necesita recaudar el proyecto.
- `country`: sigla del país de donde proviene el proyecto.
- `duration_days`: cantidad de días desde que se lanzó el proyecto hasta su fecha de límite (*deadline*).
- `state`: resultado final del proyecto. Este puede ser *failed* o *successful*. Esta es la columna que se busca predecir utilizando los demás datos.

Para esta ocasión, se harán entrega de 6 archivos `.csv`.

- *train_big.csv*: dataset que contiene todos los datos para entrenar el modelo.
- *train_medium.csv*: dataset que cuenta con el 50% de los datos para entrenar el modelo. Se mantuvo la proporción de clases cuando se generó este dataset más pequeño.
- *train_small.csv*: dataset que cuenta con el 10% de los datos para entrenar el modelo. Se mantuvo la proporción de clases cuando se generó este dataset aún más pequeño.

- *test_big.csv*: dataset que contiene todos los datos para testear el modelo.
- *test_medium.csv*: dataset que cuenta con el 50% de los datos para testear el modelo. Se mantuvo la proporción de clases cuando se generó este dataset más pequeño.
- *test_small.csv*: dataset que cuenta con el 10% de los datos para testear el modelo. Se mantuvo la proporción de clases cuando se generó este dataset aún más pequeño.

Los 6 archivos están disponible para bajar en el siguiente zip: LINK AL ZIP.

Se espera que en el punto **3** de las actividades utilicen el dataset de *train* (cualquiera de los 3) para entrenar sus modelos de clasificación y utilicen el dataset de *test* (cualquiera de los 3) para reportar los resultados. Tienes toda la libertad de dividir más el dataset de *train* para obtener un subconjunto de validación y así determinar mejores hiper-parámetros[1] Solo deben asegurarse que los archivos de *test* son únicamente para reportar los resultados finales al momento de comparar diferentes modelos y no para tomar decisiones de cómo mejorar el modelo. Finalmente, **no debes subir la base de datos a tu repositorio** o se aplicará un descuento.

Actividades

Las tareas que deberás realizar se detallan a continuación:

1. Implementar el algoritmo Árbol de Decisión

Deberán implementar una estructura de árbol de decisión, en donde ustedes empezarán con un nodo raíz que contiene a todos sus datos y cada nodo dividirá sus datos de entrada en función de 1 o más atributos, a modo de minimizar la entropía de cada subconjunto de datos generado debido a la partición.
Un ejemplo visual de un árbol de decisión es el mostrado en la figura 2.

1.1 Entropía

Para una división cualquiera de un nodo, la entropía de esta división esta dada por:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \times log\big(P(x_i)\big)$$

[1] Aunque no es requisito realizar este paso.

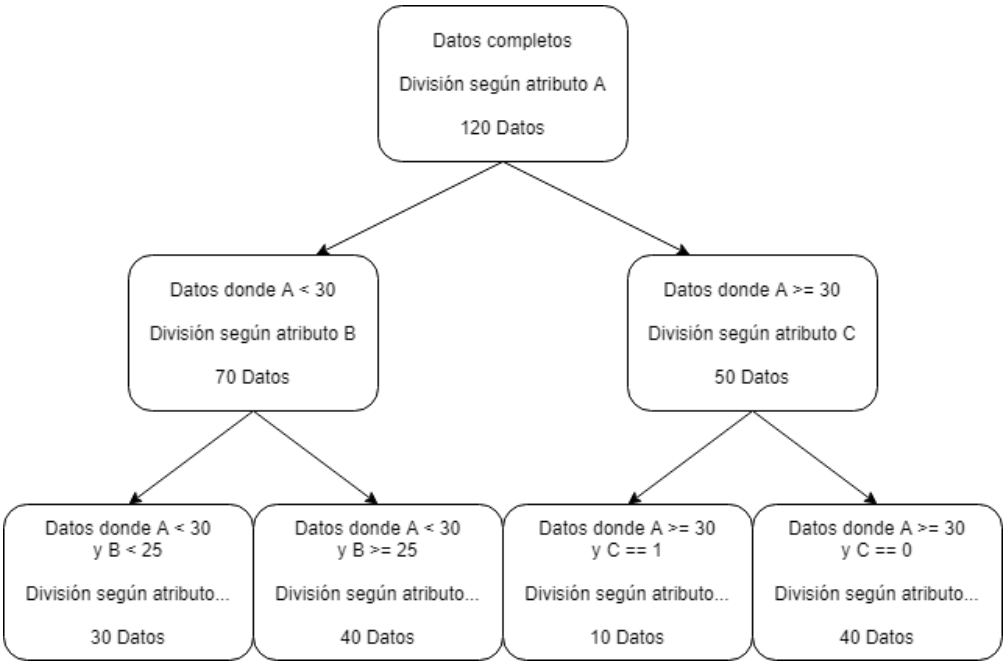

Figure 1: Visualización de árbol simple

Donde x_i corresponde al subconjunto de todos los datos cuyo valor de decisión sea el mismo y $P(x_i)$ corresponde a la fracción de todos los datos que corresponden a este subconjunto.
Por ejemplo, si tenemos 100 datos correspondientes al clima, de los cuales 45 corresponden a clima soleado x_1, 25 a clima nublado x_2 y los restantes a clima lluvioso x_3, la entropía de este conjunto es:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \times log\big(P(x_i)\big) = -\Big(\frac{45}{100} log(\frac{45}{100}) + \frac{25}{100} log(\frac{25}{100}) + \frac{30}{100} log(\frac{30}{100})\Big)$$

$$H(X) = 0.46343$$

Es posible ver con esta función, por ejemplo que si todos los datos son iguales, obtenemos una entropía cero, mientras que si cada uno fuera distinto la entropía sería de 1.

1.2 *information gain*

Para ver que tan buena es una división en específico, podrán usar la métrica llamada ***information gain*** o Ganancia de Información, la cual se obtiene con la fórmula:

$$IG(X, a) = H(X) - H(X|a)$$

Donde $H(X)$ es la entropía inicial del nodo padre (donde estamos haciendo la división) y $H(X|a)$ es la suma de las entropías de los nodos hijos dados por la separación a, ponderados por el tamaño del dataset resultante de cada hijo.
Siguiendo el ejemplo anterior, digamos que existe un criterio k que separa nuestros datos en 2 subconjuntos, uno con todos los datos de clima soleado X_1 y otro con el resto de los datos, X_2. En ese caso es *information gain* es igual a:

$$IG(X, k) = H(X) - H(X|a)$$

$$H(X|a) = \frac{|X_1|}{|X|}H(X_1) + \frac{|X_2|}{|X|}H(X_2) = 0.16458$$

Por lo tanto, el *information gain* para esta separación es

$$IG(X, a) = 0.29885$$

El *information gain* nos permite comparar entonces varios criterios para dividir los datos y elegir que criterio es el que aporta mayor cantidad de información.

1.3 Binning

Debido a que nosotros les hacemos entregas de datos continuos, para crear las divisiones deberán implementar una suerte de **data binning** en ellos, es decir que deberán agrupar los datos en diferentes "bins" para separarlos.
Las estrategias que pueden usar para la separación de los datos son las siguientes:

- Mediana: separar los datos entre los que son mayores a la mediana y los menores.
- Moda: separar los datos entre los que son iguales a la moda y los que no.
- Media aritmética: Similar a la separación anterior pero usando la media en vez de la mediana.
- Separación por Cuartiles: separar los datos en 4 subconjuntos de tamaño relativamente similar.
- Separación por Deciles: separar en 10 subconjuntos similares.

De las estrategias anteriores, usted debe escoger 2 e implementarlas en su árbol de decisión. Antes de elegir sus 2 opciones, se recomienda fuertemente leer la sección de Variaciones del algoritmo, puesto que ahí se solicita un tipo de árbol específico que puede tomar mayor tiempo en entrenar dependiendo de la estrategia de separación elegida.
Se debe implementar el algoritmo según lo visto en clases y ayudantía, **solo** podrás utilizar las librerías numpy, math, scipy, pandas e itertools.

2. Variaciones del algoritmo

Para esta parte de la tarea, deberán permitir que su árbol de decisión puede tener diferentes comportamientos al momento de generar las divisiones de cada nodo. En particular el árbol se puede configurar para que haga los cortes tradiciones, como los vistos en clases y ayudantía, o los cortes transversales. También debe ser posible indicar una máxima profundidad del árbol y la cantidad de datos mínimos que debe recibir un nodo para continuar la división.

2.1 Cortes transversales

Un corte transversal consiste en utilizar 2 atributos simultáneamente para la división de un nodo en vez de solo 1 atributo, y generar tantas divisiones como pares de divisiones posibles existan. Dado lo anterior, para obtener la mejor división con corte transversal, el árbol deberá probar todos los pares de divisiones posibles y calcular la ganancia de dicho par con el fin de obtener la que mayor ganancia entrega.

A modo de ejemplo: si un dataset tiene 2 atributos binarios (0 o 1); un corte tradicional evalúa cual de los 2 atributos entrega mayor ganancia y realiza la división por dicho atributo. Un corte transversal considera ambos atributos y genera una rama por cada tupla de divisiones posibles. Considerando el ejemplo donde son 2 atributos binarios, las divisiones que haría el nodo con cortes transversales son:

- Atributo 1 == 1 y Atributo 2 == 1.
- Atributo 1 == 1 y Atributo 2 == 0.
- Atributo 1 == 0 y Atributo 2 == 1.
- Atributo 1 == 0 y Atributo 2 == 0.

A continuación se muestran 2 ejemplos, la figura 2 corresponde a un posible árbol con cortes tradicionales, mientras que la figura 3 corresponde a un árbol con cortes transversales.
Dado lo anterior, su árbol debe recibir un argumento en el método `__init__` que puede ser `"traditional"` o `"transversal"` que indica qué tipo de corte debe realizar el árbol. Por otro lado, si se elige un árbol transversal y un nodo solo dispone de un atributo para dividir, entonces el nodo detiene su ramificación. De este modo, un árbol transversal siempre tendrá cortes que consideren 2 atributos.

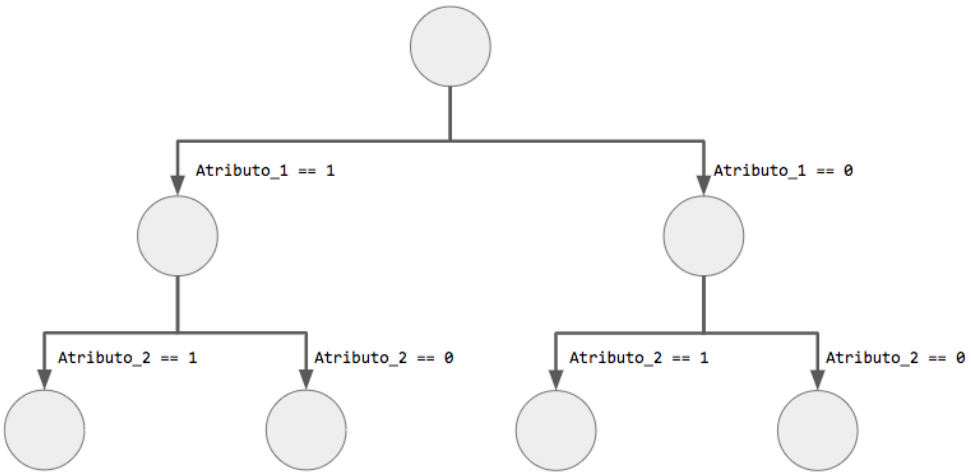

Figure 2: Visualización de árbol con cortes tradicionales

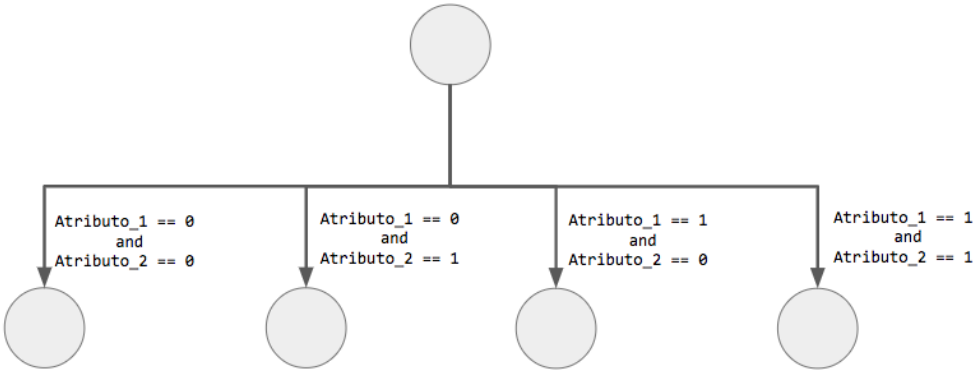

Figure 3: Visualización de árbol con cortes transversales

2.2 Definir hiper-parámetros

Adicionalmente, el árbol debe poseer 2 hiper-parámetros adicional los cuales son: máxima profundidad y mínimo de datos. Estos hiper-parámetros deben ser definidor en el método `__init__` de árbol de decisión.

1. **Máxima profundidad**: este hiper-parámetro es un número entero mayor o igual a 0 que indica la cantidad máxima de niveles que puede tener el árbol de decisión. Si un nodo llega a dicho nivel, debe detener su ramificación. Este hiper-parámetro es solo una cota máxima, el árbol no se debe condicionar a continuar formando divisiones solo para alcanzar dicha profundidad. El valor de este hiper-parámetro puede ser `None` para indicar que no hay un máximo de profundidad.

En la figura 5 se muestra un ejemplo de un árbol con el detalle de la profundidad a la que pertenece cada nodo.

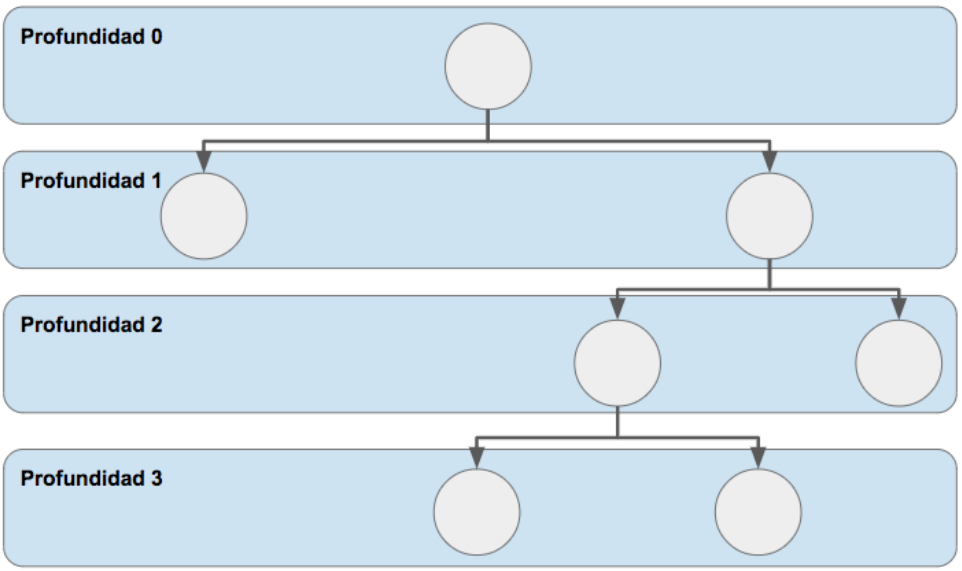

Figure 4: Profundidad a la que pertenece cada nodo del árbol

2. **Cantidad mínima de datos**: este hiper-parámetro es un número entero mayor o igual a 1 que indica cuantos datos mínimos debe poseer un nodo para continuar con su división. Si a un nodo le llegan, producto de la/s division/es anteriores, menos de esa cantidad de datos, debe detener su ramificación y considerar ese nodo como un nodo hoja. El valor de este hiper-parámetro también puede ser `None` para indicar que no hay un máximo de profundidad.

3. Comparar resultados

En esta sección tendrás que variar diferentes aspectos de la estructura del árbol, tales como la estrategia para separar los datos, los tipos de corte e hiper-parámetros con el objetivo de comparar los diferentes resultados que se obtienen y extraer conclusiones de ello. El resultado a comparar en cada sección corresponde al desempeño de árbol. Este desempeño se calcula como la división entre los datos que fueron correctamente clasificados y la cantidad total de datos clasificados. Por ejemplo, si tienen 6 datos y el árbol clasifica solo 3 de ellos de forma correcta (el algoritmo predice la clase correcta del dato), entonces el desempeño del árbol es $\frac{3}{6} = 0.5$.

3.1 Estrategias

Debes fijar los 2 hiper-parámetros como `None` y elegir un tipo de corte. Luego deberás variar la estrategia para tratar los datos según lo explicado en la sección 1.3 de Binning. Debes utiliza al menos dos separaciones distintas, compara el desempeño que obtienes con cada una y comenta.

3.2 Tipo de corte

Habiendo analizado las estrategias para el manejo de datos, quedaría ver que tan efectivos son los cortes transversales que se presentaron en la sección 2. Por lo tanto, debes fijar los 2 hiper-parámetros como `None` y elegir una estrategia de Bining. Luego deberás variar probar un árbol con cortes tradicionales (dos nodos hijos por cada nodo padre) y uno con transversales (cuatro nodos hijos) para comparar y comentar los desempeño de ambos.

3.3 hiper-parámetros

Deberá elegir la estrategia y tipo de corte que mejor resultados obtuvo de la sección 3.1 y 3.2, y ahora modificar los hiper-parámetros. En particular deberás probar 4 configuración distintas:

- El primero árbol debe tener los 2 hiper-parámetros como `None`.
- El segundo árbol debe tener la máxima profundidad como `None` y se debe modificar el hiper-parámetro de mínima cantidad de datos. Esta modificación debe generar un cambio en el rendimiento del árbol.
- El tercer árbol debe tener la mínima cantidad de datos `None` y se debe modificar el hiper-parámetro de máxima profundidad. Esta modificación debe generar un cambio en el rendimiento del árbol.
- El último árbol debe modificar ambos hiper-parámetros. Esta modificación debe generar un cambio en el rendimiento del árbol.

Se solicita experimentar con diferentes combinaciones de valores para encontrar una combinación que logre el mejor rendimiento y comenté respecto a los resultados obtenidos. ¿Cuál variación de hiper-parámetro provocó un mayor cambio en el desempeño del árbol? ¿Cambiar la máxima profundidad o cambiar la mínima cantidad de datos?.

4. Visualización del árbol

En este punto, se solicita que se pueda ver visualmente el árbol de decisión formado cuando el corte es tradicional. En esta visualización se tiene que ver como mínimo: (1) el atributo por el cual se está haciendo la división y (2) Que valor/es se hace en la división. Podrás hacer uso de todas las librerías de visualización disponibles según lo requieras. Algunos ejemplos pueden ser Graphviz, Altair, Matplotlib, Seaborn, Networkx, entre otras.

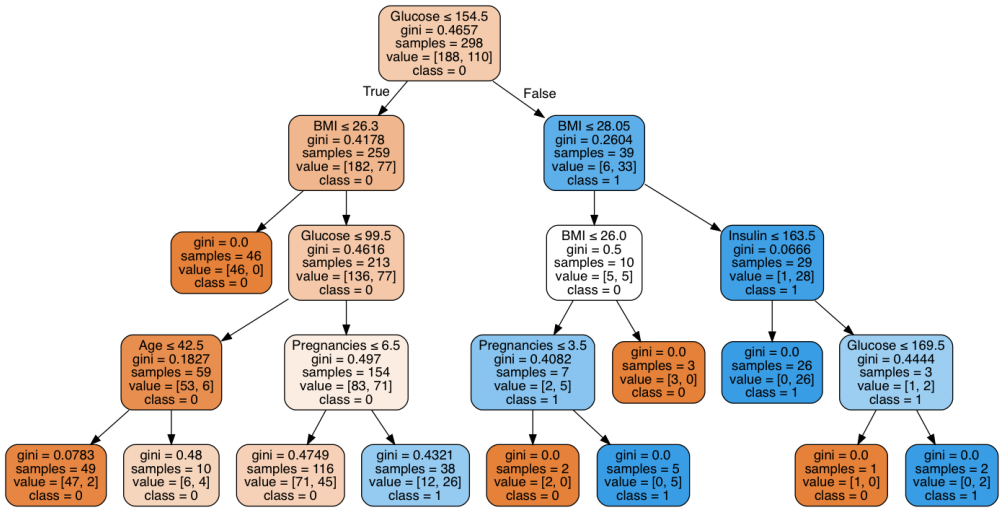

Figure 5: Visualización de un árbol utilizando *graphviz*. Extraído de DataCamp

Bonus (0.5 puntos)

Se entregará un bonus de 0.5 puntos a toda tarea que puedan visualizar un árbol con cortes trasversales.

Formato

Para esta evaluación, deberán seguir un orden específico en su .ipynb. En particular, el orden esperado es:

1. Declaración de clases: en esta sección se define la clase Node y clase DecisionTree. En particular, la clase Node debe poseer lo métodos:
    - `calculate_entropy`: método encargado de calcular la entropía de un nodo.
    - `get_best_division`: método encargado de obtener la mejor división para un nodo.

- `bining`: método encargado de categorizar una columna que disponga de datos numéricos.

Mientras que la clase DecisionTree debe poseer los métodos:

- `fit`: método encargado de entrenar el Arbol. Recibe 2 argumentos: X e Y. X corresponde al conjunto de datos para entrenar e Y es la clase a predecir.
- `predict`: método encargado de predecir los datos. Recibe 1 argumento: X que corresponde al conjunto de datos para predecir. Retorna la clase predicha para cada dato en X.
- `check_deep`: método encargado de verificar si un nodo puede seguir generando divisiones en función del hiperparámetro de máxima profundidad.
- `check_min_data`: método encargado de verificar si un nodo puede seguir generando divisiones en función del hiperparámetro de mínima cantidad de datos.

2. Comparación de resultados: en esta sección deberán realizar las 3 comparaciones, cada una en una sub-sección distinta.

3. Visualización del árbol: en esta sección se hace el llamado a la función, método o librería encargada de crear la visualización del árbol con corte tradicional. Esta visualización se debe mostrar en la celda. Por lo tanto, en caso de guardar la imagen de forma local (png, jpeg, svg, etc), debe haber una celda encargada de cargar la imagen y mostrarla en pantalla.

4. Bonus: en esta sección se hace el llamado a la función, método o librería encargada de crear la visualización del árbol con corte transversal. Esta visualización se debe mostrar en la celda. Por lo tanto, en caso de guardar la imagen de forma local (png, jpeg, svg, etc), debe haber una celda encargada de cargar la imagen y mostrarla en pantalla.

Por otro lado, deberá documentar toda función o nuevo método que haga en las clases Node y DecisionTree. Finalmente se hará entrega de un google colab con el formato que deben seguir. Dentro de cada sección pueden agregar las celdas de texto y código que consideren necesario, pero se debe respetar el orden establecido.

Entrega

Debes entregar en tu repositorio **sólo un archivo *.ipynb*** con el código y la documentación de las nuevas funciones que agreguen. Tambien puedes entretar un *Readme.md* y/o un *.gitignore* de ser necesario. **Ningún otro archivo**. Debes **seguir**

el orden especificadas en la sección Formato. Se aplicarán descuentos si no hay documentación o se entrega un informe desordenado.

Nota: Aprovecha el modo texto de *jupyter notebook* para entregar un informe más ordenado.

Evaluación

- **(0.25 pts)** Presentación y uso de *dataset*: correcta ortografía, correcta redacción en los comentarios y uso correcto de los *datasets* para entrenar y reportar resultados.
- **(0.25 pts)** Correcta implementación del método *calculate_entropy*.
- **(0.50 pts)** Correcta implementación del método *get_best_division*.
- **(0.25 pts)** Correcta implementación del método *bining*.
- **(0.50 pts)** Correcta implementación del método *fit*.
- **(0.50 pts)** Correcta implementación del método *predict*.
- **(0.25 pts)** Se compara y comenta el desempeño entre 2 estrategias de bining.
- **(1.00 pts)** Correcta implementación de los cortes trasversales.
- **(0.25 pts)** Se compara y comenta el desempeño entre los 2 cortes implementados.
- **(1.00 pts)** Correcta implementación de los 2 hiper-parámetros para podar el árbol.
- **(0.25 pts)** Se compara y comenta el desempeño entre las 4 configuraciones del árbol donde se varía los 2 hiper-parámetros.
- **(1.00 pts)** Se visualiza correctamente un árbol de corte tradicional.
- **(0.50 pts)** Bonus: Se visualiza correctamente un árbol de corte transversal.

Descuentos

Se aplicarán los siguientes descuentos:

- 0.5 puntos por no documentar nuevas funciones o métodos que implemente el alumno.
- 0.5 puntos por no respetar el formato del .ipynb entregado.
- 0.5 puntos por subir algún archivo distinto al *.ipynb, Readme.md* o *.gitignore*.