UNIDAD TEMÁTICA 4: Algoritmos Lineales PARTE 1

PRÁCTICOS DOMICILIARIOS INDIVIDUALES #1

Ejercicio 1 Revisión de los operadores disponibles en RapidMiner, Weka, AzureML, KNIME para árboles de decisión

1. Instalar y acceder a las diferentes plataformas

- RapidMiner
- Weka
- Azure Machine Learning Studio
- KNIME
- Python Scikit Learn

2. En cada plataforma, investigar los operadores o funciones de Arboles de Decisión disponibles, indicando

a. Tipos de problemas a que se pueden aplicar (clasificación o regresión, ejemplos)

b. Algoritmos de base que utilizan – breve descripción – eventualmente referencia a descripción del algoritmo y sus requerimientos y restricciones

c. Características requeridas a los atributos y a la variable objetivo

d. Parámetros que aceptan, significado y opciones disponibles

En el caso de *Weka*, encontramos que los operadores relacionados con Árboles de Decisión se encuentran principalmente a través de la implementación de algoritmos específicos. Los dos algoritmos más destacados son J48 y RandomTree, los cuales se destacan por su aplicación en datasets de clasificación, pero no se consideran apropiados para abordar problemas de regresión. Los dos algorimos se describen a continuación:

J48 (C4.5 en Weka):

El algoritmo J48 en Weka es una implementación del algoritmo C4.5. Es ampliamente utilizado para problemas de clasificación. Puedes aplicarlo a conjuntos de datos donde se requiere predecir la clase o categoría a la que pertenecen los ejemplos. Ejemplos de problemas de clasificación incluyen la detección de spam en correos electrónicos, la clasificación de especies de flores, la segmentación de clientes en categorías de comportamiento, entre otros.

RandomTree:

El algoritmo RandomTree en Weka genera Árboles de Decisión aleatorios y se utiliza principalmente para problemas de clasificación. Los árboles generados por este algoritmo son aleatorios y no necesariamente los más óptimos, pero pueden ser útiles en problemas donde se busca diversidad en la construcción del modelo. Al igual que J48, se aplica a problemas de clasificación, como la detección de spam, la clasificación de productos en categorías, etc.

Operadores:

- DecisionStump
- HoeffdingTree
- J48
- LMT
- MST
- RandomTree
- RandomForest
- REPTree

En *Azure Machine Learning Studio*, contiene un módulo "Entrenar modelo de árbol de decisión" para crear modelos de Árboles de Decisión. Los modelos resultante se pueden aplicar a problemas de clasificación y de regresión, dependiendo de cómo se haya configurado el módulo y el tipo de datos con los que analizaremos. Algunos de los casos que se pueden aplicar los Árboles de Decisión en Azure Machine Learning Studio:

Clasificación:

Los Árboles de Decisión en Azure Machine Learning Studio se pueden utilizar para problemas de clasificación, donde el objetivo es predecir la categoría o etiqueta a la que pertenece un objeto. Ejemplos de problemas de clasificación incluyen:

- Detección de spam en correos electrónicos (clasificar correos electrónicos como spam o no spam).
- Diagnóstico médico (clasificar pacientes en diferentes categorías de enfermedades).
- Segmentación de clientes (clasificar clientes en grupos según su comportamiento de compra).

Regresión:

Además de la clasificación, los Árboles de Decisión en Azure Machine Learning Studio también se pueden aplicar a problemas de regresión. En este caso, el objetivo es predecir un valor numérico en lugar de una categoría. Ejemplos de problemas de regresión incluyen:

- Predicción de precios de viviendas (predecir el precio de una casa en función de sus atributos).
- Estimación de ventas futuras (predecir las ventas de productos o servicios en función de diversas variables).
- Previsión de la demanda de productos (predecir la cantidad de productos que se venderán en el futuro).

Operadores:

Clasificación

- Multiclass Decision Forest
- Two-Class Decision Forest
- Multiclass Decision Jungle
- Two-Class Decision Jungle
- Two-Class Boosted Decision Tree

Regression

- Boosted Decision Tree Regression
- Decision Forest Regression

KNIME es una plataforma de código abierto para análisis de datos, la ciencia de datos y la automatización de procesos. Permite realizar varias tareas de análisis de datos, incluida la creación y evaluación de modelos de aprendizaje automático, como árboles de decisión. También proporciona una serie de operadores o funciones para trabajar con árboles de decisión, y estos operadores se utilizan en contextos de clasificación y regresión. Estos son algunos casos:

Árbol de Decisión (Decision Tree):

Tipo de Problema: Clasificación y regresión.

Descripción: KNIME incluye operadores para crear árboles de decisión tanto para tareas de clasificación como de regresión. Los árboles de decisión son modelos supervisados que se utilizan para predecir una variable objetivo basada en múltiples variables predictoras.

Árbol de Decisión Aleatorio (Random Forest):

Tipo de Problema: Clasificación y regresión.

Descripción: KNIME ofrece operadores para construir modelos de bosques aleatorios, que son conjuntos de árboles de decisión. Los bosques aleatorios son eficaces tanto en tareas de clasificación como de regresión, ya que reducen el sobreajuste y aumentan la precisión.

Gradiente de Árboles Potenciados (Gradient Boosting Trees):

Tipo de Problema: Clasificación y regresión.

Descripción: KNIME proporciona operadores para crear modelos de gradiente de árboles potenciados, como XGBoost y LightGBM. Estos modelos son ampliamente utilizados en competencias de ciencia de datos y son efectivos para clasificación y regresión.

Árboles de Decisión C5.0 y C4.5:

Tipo de Problema: Clasificación.

Descripción: KNIME permite crear modelos de árboles de decisión C5.0 y C4.5, que son específicos para problemas de clasificación. Estos árboles se utilizan para dividir los datos en clases o categorías.

Operadores:

- Decision Tree Learner
- Decision Tree Predictor
- Decision Tree to Image
- Decision Tree to Ruleset
- Decision Tree View (JavaScript)
- Gradient Boosted Trees Learner
- Gradient Boosted Trees Predictor

Scikit-Learn es una biblioteca de Python que ofrece una implementación eficiente de árboles de decisión tanto para problemas de clasificación como para problemas de regresión.

A continuación, se presentan los operadores y funciones disponibles en Scikit-Learn para trabajar con árboles de decisión:

DecisionTreeClassifier: Esta clase se utiliza para crear modelos de árboles de decisión para problemas de clasificación. Se pueden configurar diversos hiperparámetros, como la profundidad máxima del árbol, la cantidad mínima de muestras necesarias en un nodo para dividirlo, entre otros.

Ejemplo de clasificación:

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
```

DecisionTreeRegressor: Esta clase se utiliza para crear modelos de árboles de decisión para problemas de regresión. Al igual que en el caso de la clasificación, se pueden configurar hiperparámetros como la profundidad máxima del árbol y la cantidad mínima de muestras por nodo.

Ejemplo de regresión:

```
from sklearn.tree import DecisionTreeRegressor
reg = DecisionTreeRegressor()
reg.fit(X_train, y_train)
```

Visualización de Árboles: Scikit-Learn proporciona herramientas para visualizar árboles de decisión utilizando la función plot_tree en la subclase tree. Esto puede ayudar a comprender cómo el árbol toma decisiones basadas en las características de entrada.

Ejemplo de visualización de un árbol:

```
from sklearn.tree import plot_tree
plot_tree(clf, filled=True, feature_names=feature_names,
class_names=class_names)
```

*Predicciones: *Una vez que se ha entrenado un modelo de árbol de decisión, se pueden realizar predicciones utilizando el método predict para clasificación o predict para regresión.

Ejemplo de predicción:

```
y_pred = clf.predict(X_test)
```

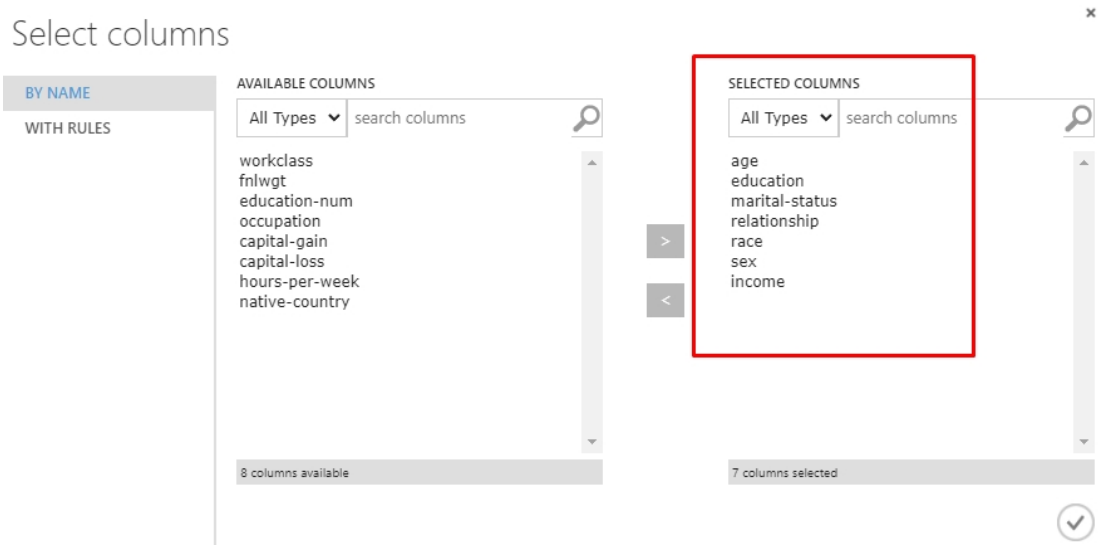
Los modelos obtenidos con Scikit-Learn permiten desarrollar los problemas del mismo modo que en las otras plataformas, con la diferencia de que en este caso se debe programar el código para cada paso del proceso.

3. Elegir datasets de práctica y resolver el problema abordado (de clasificación o regresión) con al menos 2 (dos) plataformas diferentes, compilando los resultados obtenidos.

Utilizando el Azure Machine Learning Studio, se estudio el dataset de "Adult Census Income Binary Classification dataset" (dataset ejemplo del propio azure). El objetivo es predecir si un individuo gana más de 50.000 dólares al año. El dataset contiene 32.561 registros y 15 atributos. Los atributos son los siguientes:

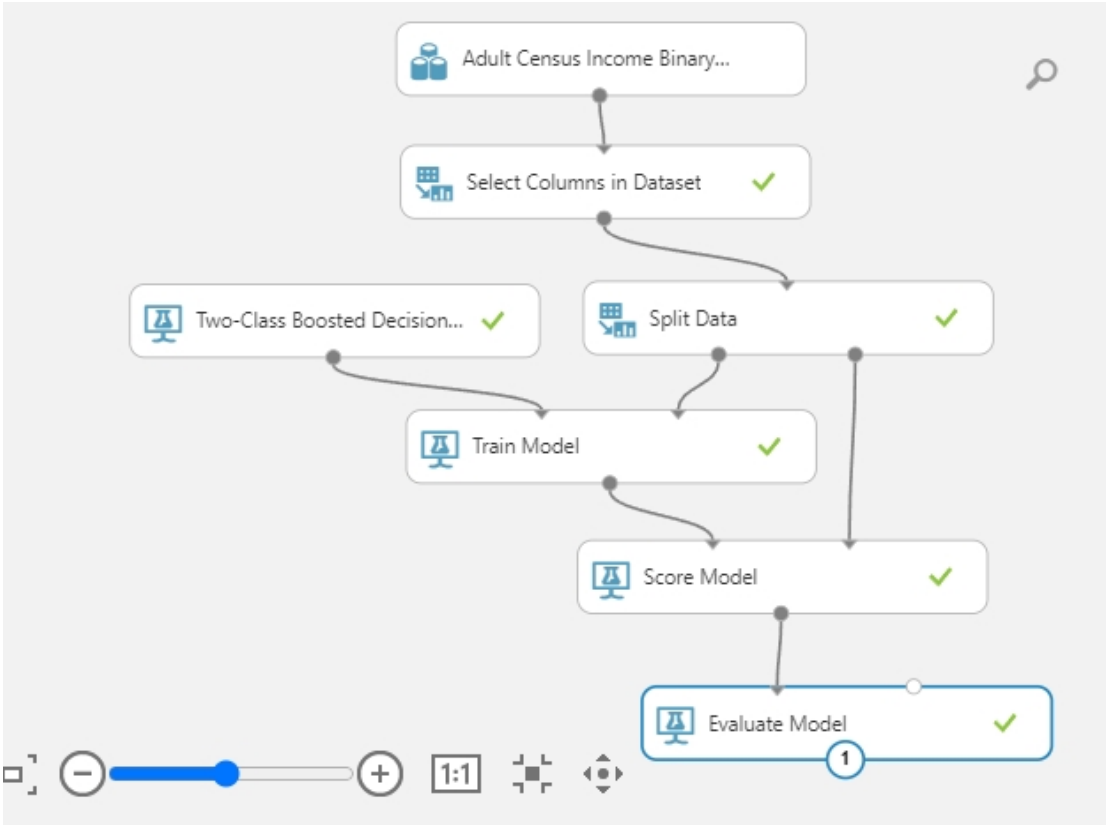
- age: Edad del individuo.
- workclass: Clase de trabajo del individuo.
- fnlwgt: Peso final, que es el número de personas que representa la entrada del censo.
- education: Nivel de educación del individuo.
- education-num: Nivel de educación del individuo, codificado como número.
- marital-status: Estado civil del individuo.
- occupation: Ocupación del individuo.
- relationship: Relación del individuo.
- race: color de la persona
- sex: Género de la persona.
- capital-gain: Ganancia de capital del individuo.
- capital-loss: Pérdida de capital del individuo.
- hours-per-week: Horas de trabajo por semana del individuo.
- native-country: País de origen del individuo.
- income: Ingresos del individuo.

Se seleccionan los atributos con mayor relevancia para realizar el estudio:

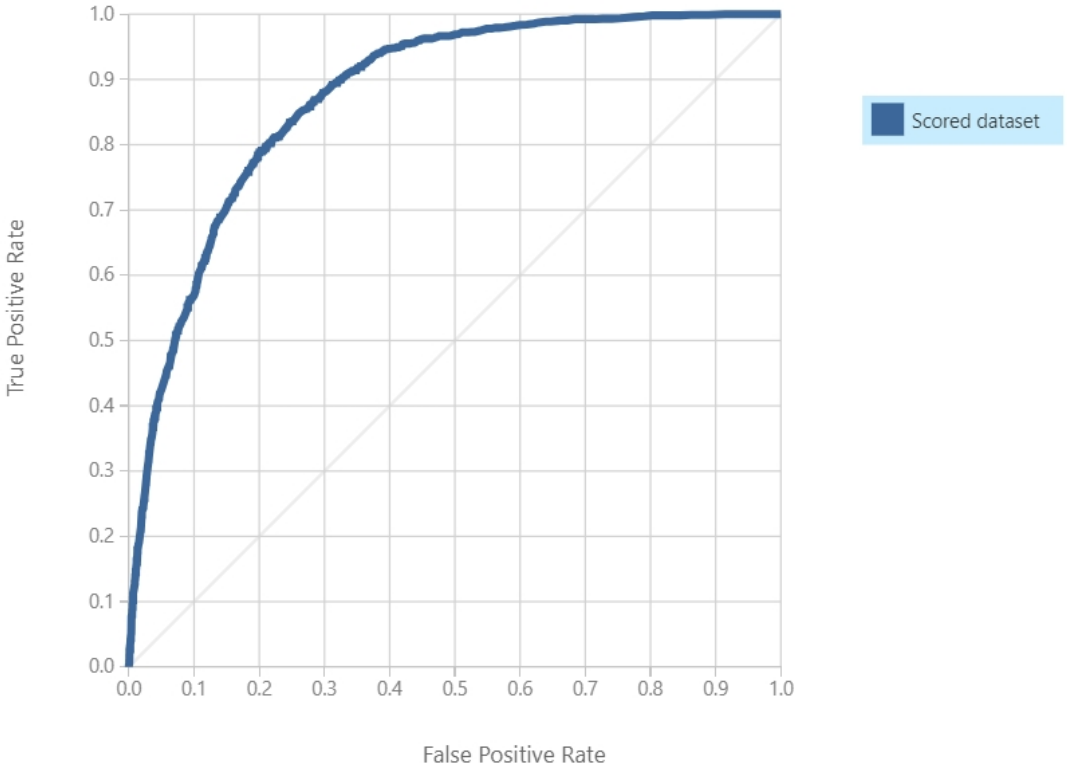


Luego se procede a realizar el modelo de árbol de decisión, con los siguientes parámetros:

- Split Data: en un 80% para entrenamiento y 20% para prueba.
- Two-Class Boosted Decision Tree: se selecciona este algoritmo para realizar el modelo de árbol de decisión.
  - Create Trainer Mode: se selecciona "Single Parameter" para poder configurar los parámetros del algoritmo.
  - Maximum number of leaves per tree: se selecciona 20 como máximo número de hojas por árbol.
  - Minimum number of samples per leaf: se selecciona 10 como mínimo número de muestras por hoja.
  - Learning rate: se selecciona 0.2 como tasa de aprendizaje.
  - Number of trees: se selecciona 100 como número de árboles.
- Train Model: se entrena el modelo con los datos de entrenamiento.



Se obtiene el siguiente modelo:



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
887	691	0.823	0.658	0.5	0.875

False Positive	True Negative	Recall	F1 Score
462	4472	0.562	0.606

Positive Label	Negative Label
>50K	<=50K

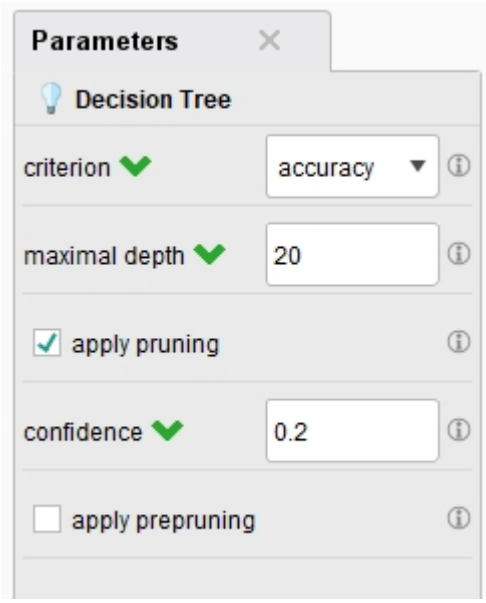
Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	43	10	0.008	0.763	0.053	0.811	0.027	0.762	0.998	0.000
(0.800,0.900]	181	40	0.042	0.784	0.242	0.818	0.142	0.783	0.990	0.001
(0.700,0.800]	307	110	0.106	0.815	0.468	0.768	0.337	0.820	0.968	0.006
(0.600,0.700]	135	80	0.139	0.823	0.536	0.735	0.422	0.837	0.951	0.012
(0.500,0.600]	221	222	0.207	0.823	0.606	0.658	0.562	0.866	0.906	0.035
(0.400,0.500]	177	186	0.263	0.822	0.647	0.621	0.674	0.893	0.869	0.058
(0.300,0.400]	180	345	0.344	0.796	0.652	0.556	0.788	0.922	0.799	0.109
(0.200,0.300]	118	395	0.422	0.754	0.629	0.495	0.863	0.943	0.719	0.175
(0.100,0.200]	115	462	0.511	0.700	0.602	0.444	0.936	0.968	0.625	0.259
(0.000,0.100]	101	3084	1.000	0.242	0.390	0.242	1.000	1.000	0.000	0.875

RapidMiner

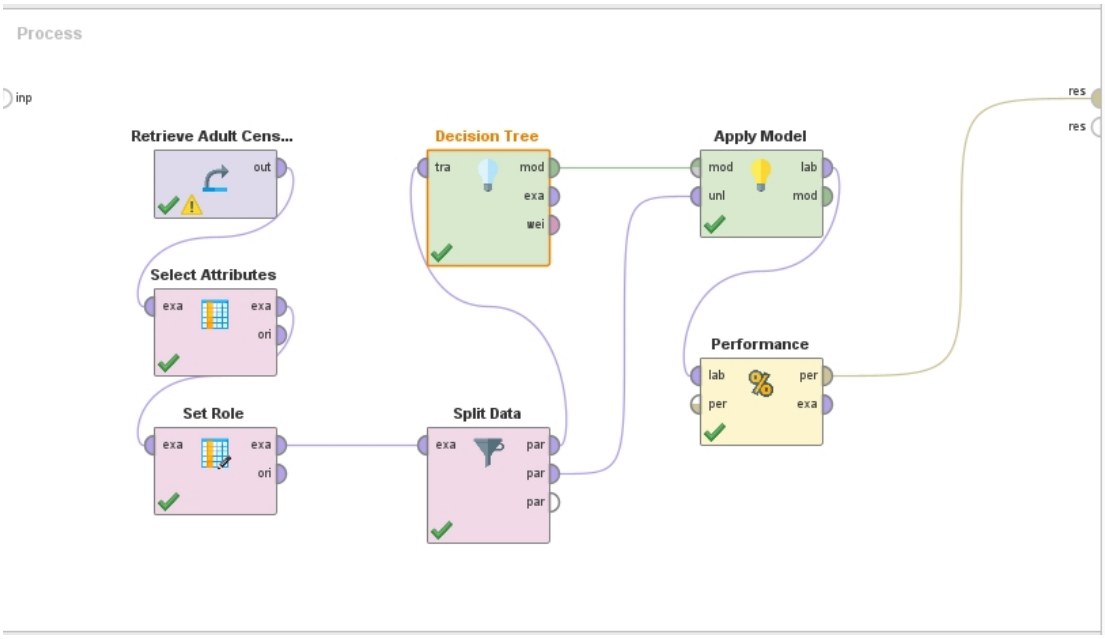
Se utilizó el mismo dataset de "Adult Census Income Binary Classification dataset" para realizar el modelo de árbol de decisión en RapidMiner. Se seleccionan los atributos con mayor relevancia para realizar el estudio al igual que en Azure Machine Learning Studio:

- age
- education
- marital-status
- race
- relationship
- sex
- income

Se procede a realizar el modelo de árbol de decisión, con los siguientes parámetros:



Se obtiene el siguiente modelo:



Y los resultados correspondientes fueron los siguientes:

Table View Plot View

accuracy: 82.11%

	true <=50K	true >50K	class precision
pred. <=50K	4641	862	84.34%
pred. >50K	303	706	69.97%
class recall	93.87%	45.03%	

Row No.	income	prediction(in...	confidence(...	confidence(...	age
1	<=50K	<=50K	0.837	0.163	39
2	<=50K	>50K	0.298	0.702	50
3	<=50K	>50K	0.268	0.732	28
4	<=50K	>50K	0.217	0.783	37
5	>50K	>50K	0.298	0.702	30
6	<=50K	<=50K	0.841	0.159	32
7	<=50K	>50K	0.413	0.587	48
8	<=50K	<=50K	0.640	0.360	24
9	<=50K	<=50K	0.647	0.353	44
10	<=50K	>50K	0.298	0.702	30
11	<=50K	<=50K	0.841	0.159	36
12	<=50K	<=50K	0.963	0.037	31
13	>50K	>50K	0.298	0.702	29

4. Conclusiones

En ambos casos los resultados obtenidos fueron similares ya que se trata de una clasificación de un mismo dataset con los mismos atributos y selecciones. Aún así el estudio realizado en Azure Machine Learning Studio fue más sencillo y rápido de realizar, ya que la plataforma es más intuitiva y fácil de utilizar. En cambio, RapidMiner requiere de un mayor conocimiento de la plataforma para poder realizar el estudio de manera eficiente.

Otro punto importante es que Azure Machine Learning Studio realiza el trabajo en la nube, por lo cual el equipo que posea el usuario no es una limitante para poder desarrollar algoritmos complejos, ya que el poder de procesamiento esta dado por la propia plataforma. En cambio, RapidMiner requiere de un equipo con un buen poder de procesamiento para poder realizar los algoritmos de manera eficiente.