Aprendizaje Automático I

1 PARCIAL
1° CUATRIMESTRE, 2024
LIC. JAVIER DI SALVO

GRUPO 6

MORENO PRISCILA

FEDERICO GAUNA

JUAN TOROSSI

MARIA BARRETO

CONSIGNA INDIVIDUAL

SE UTILIZA UNA BASE DE DATOS QUE CONTIENE INFORMACIÓN SOBRE INDICADORES SOCIOECONÓMICOS EN LA REPÚBLICA ARGENTINA.
(TABLA_GDP_1PARCIAL.XLSX).

DESARROLLAR UN ALGORITMO QUE PERMITA ESTIMAR EL NIVEL DEL ANALFABETISMO EN RELACIÓN A LA POBREZA Y A LA DESERCIÓN ESCOLAR EN LAS DISTINTAS PROVINCIAS DE LA ARGENTINA.

Análisis de Indicadores Socioeconómicos en Argentina: Estimación del Nivel de Analfabetismo

Dataset

Primera observación de los datos proporcionados sobre indicadores socioeconómicos en la República Argentina. La tabla proporcionada consta de **11 columnas,** cada una representa un indicador específico relacionado con la situación socioeconómica en las provincias argentinas.

La tabla contiene datos relevantes para el desarrollo de un algoritmo destinado a estimar el **nivel de analfabetismo en relación con la pobreza y la deserción escolar** en las distintas provincias de Argentina. Las columnas pertinentes para cumplir con la consigna serían:

- **Illiteracy: Tasa de analfabetismo en la provincia.**
- **Poverty: Porcentaje de población en situación de pobreza.**
- **School Dropout: Tasa de deserción escolar en la provincia.**

Visualización de los primeros cinco registros del dataset.

	province	gdp	illiteracy	poverty	deficient_infra	school_dropout	no_healthcare	birth_mortal	pop	movie_theatres_per_cap	doctors_per_cap
0	Buenos Aires	2.926899e+08	-1.038651	-0.476036	5.511856	-0.881165	48.7947	4.4	15625084	0.000006	0.004836
1	Catamarca	6.150949e+06	-0.478949	-0.187273	10.464484	-0.714726	45.0456	1.5	367828	0.000005	0.004502
2	Córdoba	6.936374e+07	-0.263433	-1.230354	10.436086	-0.640290	45.7640	4.8	3308876	0.000011	0.010175
3	Corrientes	7.968013e+06	0.000000	0.764107	17.438858	1.894176	62.1103	5.9	992595	0.000004	0.004495
4	Chaco	9.832643e+06	2.534459	1.607794	31.479527	0.741431	65.5104	7.5	1055259	0.000003	0.003605

Visualización de los primeros diez registros de las variables que se utilizarán en el modelo.

	illiteracy	poverty	school_dropout
0	1.383240	8.167798	0.766168
1	2.344140	9.234095	0.951963
2	2.714140	5.382380	1.035056
3	NaN	12.747191	3.864265
4	7.517580	15.862619	2.577462
5	1.548060	8.051752	0.586309
6	3.185580	7.288751	NaN
7	4.610640	17.035583	2.268974
8	2.151390	13.367965	0.721295
9	1.539300	3.398774	0.204093

Preprocesamiento

La etapa de preprocesamiento de datos es crucial para garantizar que los datos estén limpios y listos para utilizarlos en el modelo predictivo. Primero identificamos datos faltantes, obtenemos como resultado que las variables “gdp”, “illiteracy”, “school_dropout” y “no_healthcare” tienen 2 valores nulos cada una.

```
df.isnull().sum()
```

```
province                0
gdp                     2
illiteracy              2
poverty                 0
deficient_infra         0
school_dropout          2
no_healthcare           2
birth_mortal            0
pop                     0
movie_theatres_per_cap  0
doctors_per_cap         0
dtype: int64
```

Revisamos el tipo de dato de estas variables.

```
df.dtypes
```

```
province                 object
gdp                     float64
illiteracy              float64
poverty                 float64
deficient_infra         float64
school_dropout          float64
no_healthcare           float64
birth_mortal            float64
pop                       int64
movie_theatres_per_cap  float64
doctors_per_cap         float64
dtype: object
```

Para resolver esto, rellenamos estos espacios con el valor medio de cada uno de estos conjuntos de datos. Esta técnica de imputación permite mantener la integridad de los datos al proporcionar valores estimados para el datos faltante.

```
means = df.select_dtypes(include='float64').mean()

# Rellenar los valores nulos con la media de cada columna
df.fillna(means, inplace=True)
```

Luego realizamos la estandarización del conjunto de datos a trabajar, utilizando: **StandardScaler**. Con el propósito de que las variables tengan la misma escala, y de esa manera facilita la comparación e interpretación del modelo.

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
df[['illiteracy','poverty','school_dropout']]=scaler.fit_transform(df[['illiteracy','poverty','school_dropout']])
```

Argumentación de árbol de regresión

Para decidir el tipo de árbol que usar hay que tener en cuenta principalmente para qué estamos desarrollando este algoritmo, las variables que se van a utilizar y los tipos de algoritmos que tenemos disponibles.

Lo que se quiere conseguir es una estimación del nivel de analfabetismo en base a la pobreza y la deserción escolar, por lo cual tenemos al analfabetismo como variable dependiente y a la pobreza y la deserción escolar como variables independientes. Estas tres recién nombradas son las que van a participar en el desarrollo del modelo. Lo que tienen en común es que son todas variables numéricas, por lo cual, la variable respuesta será numérica. Y por último, los algoritmos que hay para utilizar son: árbol o bosque de clasificación y árbol o bosque de regresión. Debido a que nuestra variable respuesta es numérica lo más lógico es utilizar un árbol o bosque de regresión, cuya predicción será un valor numérico, en contraste de los de clasificación, que su respuesta es una variable categórica. Y para elegir entre un árbol o bosque, el razonamiento está en que cantidad de datos se van a trabajar. En este caso, es un dataset bastante pequeño, por lo cual las ventajas que nos da un bosque en cuanto a evitar el overfitting y la precisión de los datos, no son necesarias porque hay muy pocos datos para sacar provecho de esto. En cambio, un único árbol de regresión es suficiente para poder conseguir unos valores de predicción agradables.

Entrenamiento del modelo

Lo primero que necesitamos hacer antes de comenzar con el entrenamiento es dividir los datos en un conjunto de entrenamiento y un conjunto de testeo.

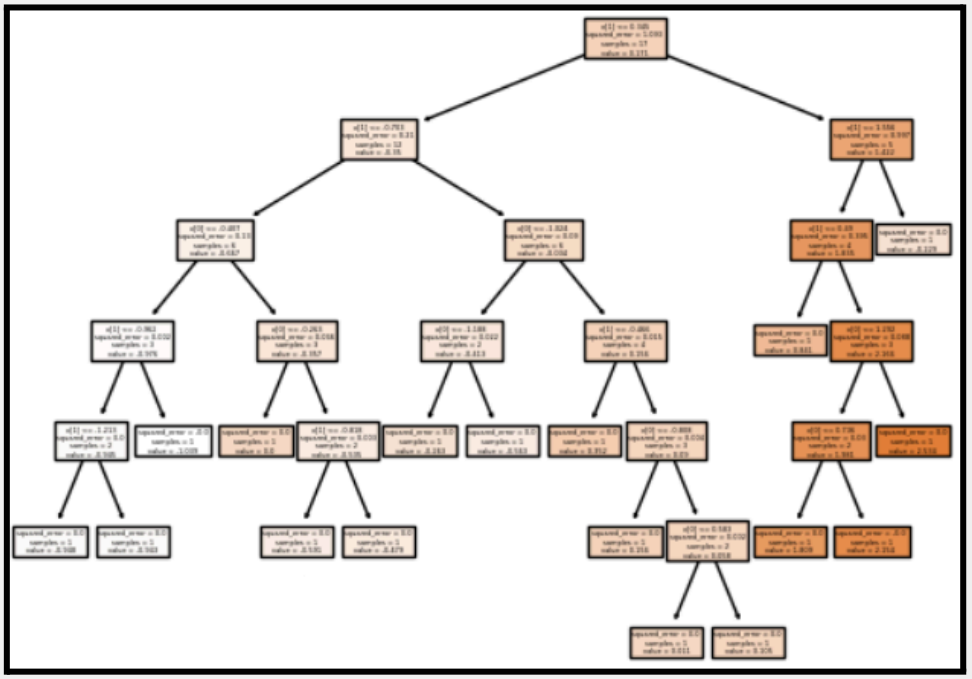
```
#2. Split de los datos en los conjuntos de entrenamiento y testeo
from sklearn.model_selection import train_test_split
x=pd.DataFrame(df,columns=['poverty','school_dropout'])      # Variables observables
y=pd.DataFrame(df,columns=['illiteracy']) # Variable Objetivo

#Particion del dataset con 20% datos para el conjunto de testing, 80% entrenamiento
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=1)
```

Las variables observables o las “x” que tenemos son “poverty” y “school_dropout”. Mientras que la variable objetivo o la “y” es “illiteracy”.

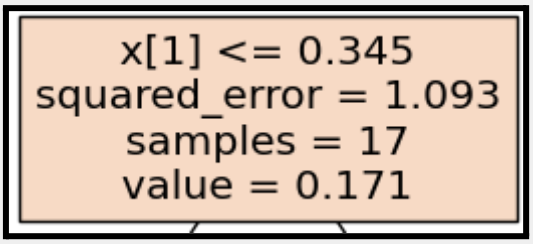
El método de partición que se utilizará es el de 80 / 20. De esta forma, garantizamos que haya una buena cantidad de datos de entrenamiento para asegurar un modelo de aprendizaje lo más preciso posible.

Una vez hecha esta división de los datos, se puede proceder a la creación del algoritmo que predecirá el valor de analfabetismo de las provincias de la Argentina en función de la pobreza y la deserción escolar.

El criterio de decisión que se utiliza en este árbol es el “squared_error”. Lo que se puede destacar, por ahora, de este árbol es que tiene 17 hojas.



En el nodo raíz se muestra la información del primer split que se hace en el árbol. Se destacan cuatro datos importantes:



- La condición que tienen que cumplir los valores de X para que pertenezcan a ese nodo. En este caso, se está evaluando a la variable “school_dropout”, la cual tiene que ser menor o igual a 0,345.

- El valor del criterio utilizado para hacer las divisiones. En este caso, es “squared_error” con un valor de 1,093.

- La cantidad de registros que se están agrupando en ese nodo. En este caso, al ser el nodo raíz, contiene todos los registros que se están utilizando en el conjunto de entrenamiento, que al ser del 80% de los datos, son un total de 17.

- ·El valor promedio de la variable objetivo. En este punto, es de 0,171.

Predicción del modelo

3. ¿Qué tipo de variables se muestran en los nodos de decisión?

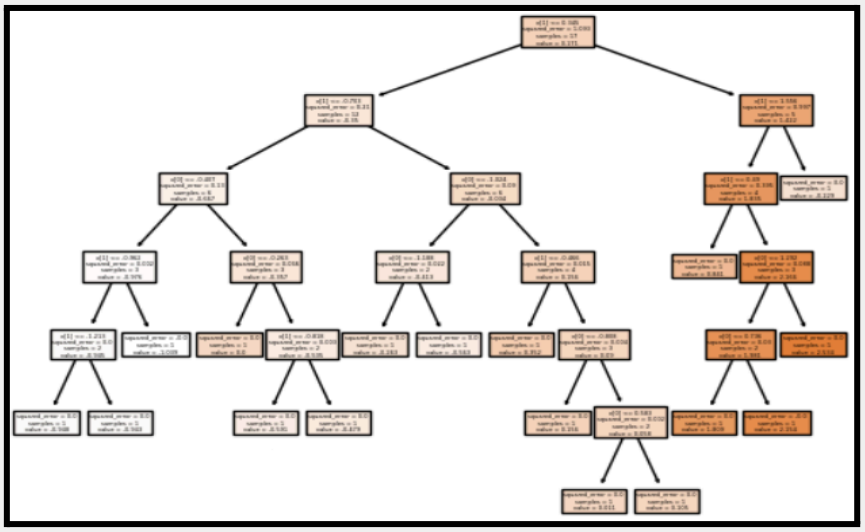
Las variables que se muestran en los nodos de decisión son las variables predictoras utilizadas para entrenar el modelo de árbol de regresión. Las variables predictoras son “poverty” y “school dropout”.

4. ¿Qué variable se representa en los nodos hojas?

En los nodos hojas del árbol de regresión se representa la variable de respuesta|variable objetivo|variable dependiente. En nuestro caso se llama “illiteracy”, es la variable que queremos predecir. Entonces cada nodo hoja representa una predicción para las observaciones que terminan en ese nodo después de seguir el camino a través del árbol desde el nodo raíz hasta el nodo hoja correspondiente.

5. ¿Cuál es el nivel de árbol sin podar?

El Árbol sin podar contiene 6 niveles.



Para efectuar la predicción de la variable respuesta utilizamos el método “predict” del modelo entrenado. Utilizamos model1 para hacer las predicciones sobre el conjunto de datos de prueba “x_test”, y el resultado de estas predicciones se almacenan en la variable “y_pred_model1”.

Luego comparamos estas predicciones con los valores reales de la variable respuesta. Creamos un DataFrame “df_prediccion” que contiene dos columnas, una para valores reales y otra para los valores predichos. Esto nos permite visualizar cómo se comparan las predicciones del modelo con los datos reales. Cada fila corresponde a una observación en

el conjunto de prueba, donde se muestra el valor real de la variable respuesta y el valor predicho por el modelo. Esto nos facilita la evaluación del rendimiento del modelo en base a la discrepancia entre los valores reales y predichos.

Predicción

```
[16] y_pred_model1=model1.predict(x_test)
     y_pred_model1
```

```
array([ 1.80899012,  1.80899012, -0.2290259 ,  0.01117001, -0.9477497 ])
```

```
d = {'valores reales': y_test['illiteracy'], 'valores predichos': y_pred_model1}
df_prediccion=pd.DataFrame(data=d)
df_prediccion
```

	valores reales	valores predichos
19	-0.693417	1.808990
16	-0.118326	1.808990
3	0.000000	-0.229026
13	-0.712167	0.011170
18	-1.383335	-0.947750

Evaluación del modelo

Una vez tenemos ya hechas las predicciones, se utilizarán métricas para evaluar la precisión del modelo. Tenemos en cuenta los valores de las siguientes métricas: MAE, MSE y RMSE. Cuanto más disminuyen estos números, mejor predicciones estará dando el modelo.

```
The mean absolte error (MAE) on test set: 1.16353
The mean squared error (MSE) on test set: 2.1484
The root mean squared error (RMSE) on test set: 1.4657
```

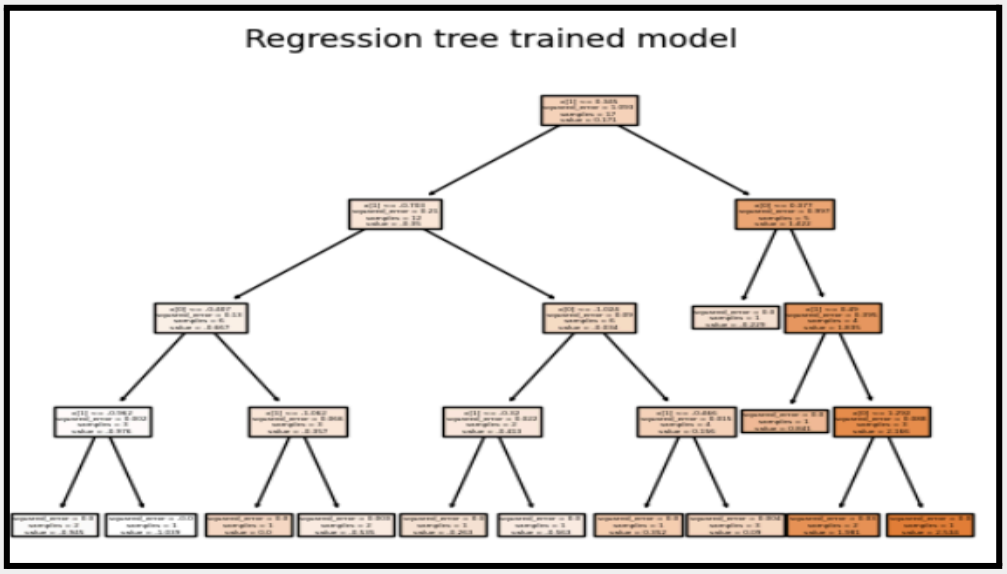
Segundo modelo (árbol podado)

6. Efectuar una poda del árbol considerando 4 niveles.

Muchas veces para encontrar la mejor calidad en nuestras predicciones es necesario hacer más de un modelo predictivo, y compararlos entre ellos para ver cuál es el que mejor resultados da. Lo que se busca cambiar en estos otros modelos son los hiper parámetros que tienen los árboles de decisión. Jugando con estos es posible llegar a distintos y mejores resultados.

En este caso, lo que se hará es otro árbol de regresión el cuál estará podado, es decir, tendrá limitado su nivel máximo. En este caso, está limitación será de cuatro niveles.

```
model2 =DecisionTreeRegressor (criterion='squared_error', random_state=1, max_depth=4)

modelTrained=model2.fit(x_train, y_train)
```



7. Efectuar la predicción de la variable respuesta.

Ya con el árbol podado, se vuelven a hacer predicciones para comparar estos valores con los ya obtenidos anteriormente con el árbol sin podar.

Predicción

```
[21] y_pred_model2=model2.predict(x_test)
     y_pred_model2
```

```
array([-0.2290259 , -0.2290259 ,  1.98136244,  0.09049945, -0.94519846])
```

```
[22] d = {'valores reales': y_test['illiteracy'], 'valores predichos': y_pred_model2}
     df_prediccion=pd.DataFrame(data=d)
     df_prediccion
```

	valores reales	valores predichos
19	-0.693417	-0.229026
16	-0.118326	-0.229026
3	0.000000	1.981362
13	-0.712167	0.090499
18	-1.383335	-0.945198

Se utilizan las mismas métricas usadas en el otro modelo para evaluar la precisión del árbol.

```
The mean absolte error (MAE) on test set: 0.75945
The mean squared error (MSE) on test set: 0.9980
The root mean squared error (RMSE) on test set: 0.9990
```

Una vez hecho todo esto, se puede pasar a la comparación entre ambos modelos para ver cual es el que da mejores valores predichos.

8. ¿Al comparar el árbol podado y el árbol sin podar cual utilizarían para realizar la predicción de la variable respuesta? Justificar en base a las métricas que se evalúan en el modelo.

Para poder decidirnos entre el Árbol Podado (Modelo 2) y el Árbol sin podar (Modelo 1) para realizar la predicción de la variable respuesta, consideramos las métricas de evaluación del modelo:

- Error Absoluto Promedio (**MAE**)
- Error Cuadrático Promedio (**MSE**)
- Raíz Cuadrada del Error Cuadrático Promedio (**RMSE**)

Comparando las métricas para ambos modelos, observamos que todas las métricas del Modelo 2 “Árbol Podado” son menores que las del Modelo 1 (Árbol sin podar). Esto nos indica que el **Modelo 2**, tiene un **mejor rendimiento** en términos de precisión en comparación con el Modelo 1. Específicamente, el MAE, MAE y RMSE son menores en el Modelo 2, lo que significa que las predicciones están **más cerca de los valores reales**.

Modelo 1 (Árbol sin podar):	Modelo 2 (Árbol podado)
MAE: 1.16353	MAE: 0.75945
MSE: 2.1484	MSE: 0.9980
RMSE: 1.4657	RMSE: 0.9990