ISTEA

Aprendizaje Automático I

Análisis para estimar el nivel de analfabetismo en relación a la deserción escolar y a la mortalidad infantil de la República Argentina

---

Se nos solicitó estimar los niveles de analfabetismo en las provincias de la República Argentina, teniendo en consideración los indicadores de deserción escolar y mortalidad infantil.
Para realizar dicha estimación, se utiliza una base de datos que contiene información sobre indicadores socioeconómicos en la República Argentina.

Se procedió a trabajar en el desarrollo de un modelo predictivo de Árbol de decisión - Random Forest Classifier, centrándonos en una variable categórica, que nos permitiera analizar los indicadores solicitados de dicha base de datos, utilizando el lenguaje de programación Python en entornos como Visual Studio Code y Google Collaboratory.

El trabajo consistió en el desarrollo de las siguientes etapas:

- Preprocesamiento
- Splitting
- Entrenamiento
- Evaluación
- Validación

Las etapas anteriormente mencionadas se desarrollarán a continuación.

Preprocesamiento

Se comenzó con la importación de las librerías necesarias para el desarrollo del modelo, a saber: Pandas, Numpy, Scikit-Learn, Matplotlib y Seaborn.
A su vez, se importaron los estimadores correspondientes a la librería de Scikit-learn, como Simple Imputer, Standard Scaler, para la realización del procesamiento.
Por último, se importaron los estimadores train_test_split, Random Forest Classifier, plot_tree y metrics como Accuracy Score, Recall Score, Precision Score y Matriz de confusión para la realización del modelo de Random Forest y métricas de evaluación.

Una vez importadas las librerías y estimadores necesarios, se continuó con la lectura de la base de datos proporcionada. Dicha base de datos muestra los índices socio-económicos, como densidad poblacional, salud, educación, pobreza, deserción escolar, analfabetismo, mortalidad infantil e infraestructura correspondientes a 22 provincias de la República Argentina. Se realizó una lectura de las estructuras descriptivas de la base de datos para tener una mejor interpretación de la misma, evaluando inconsistencias.

Para la realización de nuestro modelo predictivo, nos enfocamos en los indicadores correspondientes a la deserción escolar, mortalidad infantil y analfabetismo. Por este motivo, nos dispusimos a evaluar la existencia de valores nulos y duplicados en las columnas mencionadas. Encontramos que tanto deserción escolar como analfabetismo, poseían algunos valores nulos, por lo que decidimos reemplazarlos realizando un promedio de los valores totales correspondientes a cada columna. El dataset no poseía valores duplicados.

Cantidad de valores nulos

```
print("Valores nulos en el conjunto de datos:")
df.isnull().sum()
```

```
Valores nulos en el conjunto de datos:
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
```

Identificando si hay valores duplicados

```
[ ] hay_duplicados = df.duplicated().any()

    if hay_duplicados:
        print("Hay registros duplicados en el DataFrame.")
    else:
        print("No hay registros duplicados en el DataFrame.")
```

```
No hay registros duplicados en el DataFrame.
```

Finalmente, se realizó una copia del database en un dataframe el cual tuviera únicamente los indicadores de las columnas a utilizar. (provincia, analfabetismo, mortalidad infantil y deserción escolar).

Posteriormente, mediante la utilización de SimpleImputer, nos aseguramos de que los valores nulos quedan reemplazados por los valores de los promedios correspondientes.

```
[ ] # Remplazamos los valores nulos por la media de la columna respectiva
    imp = SimpleImputer(missing_values=np.nan, strategy='mean')
    df_copia[['illiteracy','school_dropout']]=imp.fit_transform(df_copia[['illiteracy','school_dropout']])
    df_copia
```

Una vez reemplazados los valores nulos, procedimos a corroborar que las columnas estén completas, con una respuesta exitosa.

```
print("Valores nulos en el conjunto de datos:")
df_copia.isnull().sum()
```

```
Valores nulos en el conjunto de datos:
province          0
illiteracy        0
school_dropout    0
birth_mortal      0
dtype: int64
```

Luego, se procedió a realizar la normalización del dataframe utilizando el StandardScaler.

```
[ ] # Estandarización
    scaler=StandardScaler()
    df_copia[['illiteracy','school_dropout','birth_mortal']]=scaler.fit_transform(df_copia[['illiteracy','school_dropout','birth_mortal']])
    df_copia
```

Como se nos solicitó estimar el nivel de analfabetismo, se continuó con el cálculo del primer, segundo y tercer cuartil de la columna “illiteracy” (analfabetismo). De esta forma, agregamos una nueva columna llamada “illiteracy_index”, para agregar las variables categóricas “High”, “Medium”, y “Low” para evaluar el nivel de analfabetismo considerando los indicadores de mortalidad infantil y deserción escolar.

```
[ ]  # Se calcula primer, segundo y tercer cuartil de la columna illiteracy
     q1 = np.quantile(df_copia['illiteracy'],0.25)
     q2 = np.quantile(df_copia['illiteracy'],0.50)
     q3 = np.quantile(df_copia['illiteracy'],0.75)

     print(q1)
     print(q2)
     print(q3)
```

```
-0.6678682022230867
-0.17367605340176506
0.14291458602813512
```

```
[ ]  # Creamos columna nueva
     df_copia['illiteracy_index']=''

     # Categorizamos la variable analfabetismo
     df_copia.loc[df_copia['illiteracy']<q1,'illiteracy_index'] ='Low'
     df_copia.loc[(df_copia['illiteracy']>=q1)&(df_copia['illiteracy']<q2),'illiteracy_index'] ='Middle'
     df_copia.loc[df_copia['illiteracy']>=q2,'illiteracy_index'] ='High'
     df_copia
```

El resultado de la nueva base de datos fue el siguiente:

[ ]

	province	illiteracy	school_dropout	birth_mortal	illiteracy_index
0	Buenos Aires	-1.038651	-0.881165	-0.171556	Low
1	Catamarca	-0.478949	-0.714726	-1.020029	Middle
2	Córdoba	-0.263433	-0.640290	-0.054526	Middle
3	Corrientes	0.000000	1.894176	0.267309	High
4	Chaco	2.534459	0.741431	0.735432	High
5	Chubut	-0.942647	-1.042286	-0.581164	Low
6	Entre Ríos	0.011170	0.000000	-0.551906	High
7	Formosa	0.841234	0.465081	3.280849	High
8	Jujuy	-0.591222	-0.921364	-0.376360	Middle
9	La Pampa	-0.947750	-1.384683	0.647658	Low
10	La Rioja	-0.229026	1.876873	1.876481	Middle
11	Mendoza	-0.562791	0.000000	-0.171556	Middle
12	Misiones	2.153735	1.235622	0.910978	High
13	Neuquén	-0.712167	-0.319184	-0.493391	Low
14	Río Negro	0.000000	-1.201982	-1.224833	High
15	Salta	0.104663	-0.239881	0.238051	High
16	San Juan	-0.118326	1.387274	-0.230072	High
17	San Luis	0.155665	0.224282	-0.347102	High
18	Santa Cruz	-1.383335	-1.308387	-0.493391	Low
19	Santa Fe	-0.693417	1.005446	-0.698194	Low
20	Santiago del Estero	1.808990	0.515806	-0.961514	High
21	Tucumán	0.351797	-0.692042	-0.581164	High

Splitting

Una vez terminado el preprocesamiento, se comenzó con la realización del modelo predictivo.
En esta instancia, se procedió con la partición de nuestra base de datos en un 70% para el entrenamiento (training) y un 30% para el testeo (testing).

```
0# Particion del dataset con 30% datos para el conjunto de testing, 70% entrenamiento
x=pd.DataFrame(df_copia,columns=['illiteracy','school_dropout','birth_mortal'])
y=pd.DataFrame(df_copia,columns=['illiteracy_index'])


x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3, random_state=42)
```

```
print("x_train shape:",x_train.shape)
print("y_train shape:",y_train.shape)
print("x_test shape:",x_test.shape)
print("y_test shape:",y_test.shape)
y_train
```

```
x_train shape: (15, 3)
y_train shape: (15, 1)
x_test shape: (7, 3)
y_test shape: (7, 1)
```

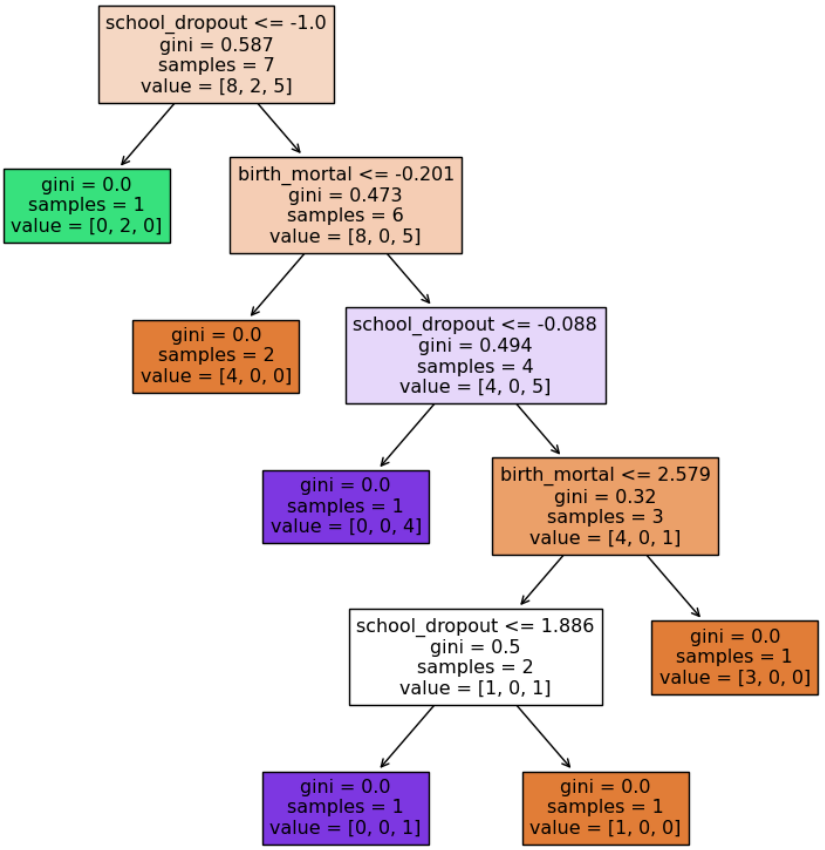
Entrenamiento y Evaluación

Se realizó un modelo de Árbol de decisión Random Forest Classifier considerándola una técnica de aprendizaje automático supervisado utilizada para la clasificación de los datos y así obtener una predicción categórica más precisa y estable.

Evaluando distintos números de estimadores, nuestro primer árbol del Random Forest Classifier quedó representada de la siguiente forma:

Sin poda:

Partición: 0.3
Estimadores: 80
Profundidad: None
Posición: 0
Accuracy: 71.43



Realizando el análisis del árbol sin podar, pudimos visualizar que el nodo raíz refleja la variable deserción escolar, con una impureza, indicada con el índice Gini del 0.587, un número de samples (muestras) de 7 y valores de [8,2,5] representando los niveles “High”, “Middle” y “Low” respectivamente.

Del nodo raíz se desprende una hoja y un nodo de decisión. El nodo de decisión (primer nivel) refleja la variable mortalidad infantil con un índice Gini de 0.473, un número de muestras de 6 y valores [8,0,5]. La hoja posee un Gini de 0, con valores de [0,2,0], representando 2 valores “Middle” puros.

A su vez, de ese nodo se desprende una hoja con un índice gini = 0, con 2 muestras y valores de [4,0,0], representando 4 valores “High”. También se desprende otro nodo de decisión que refleja la variable deserción escolar (segundo nivel), con un índice Gini de 0.494, 4 muestras y valores de [4,0,5].

De este nodo antes mencionado, se desprende una hoja con 0 índice Gini, 1 muestra y valores de [0,0,4], representando 4 valores “Low”. También se desprende otro nodo de decisión, en este caso con la variable mortalidad infantil (tercer nivel), con un índice Gini de 0.32, 3 muestras y valores de [4,0,1].

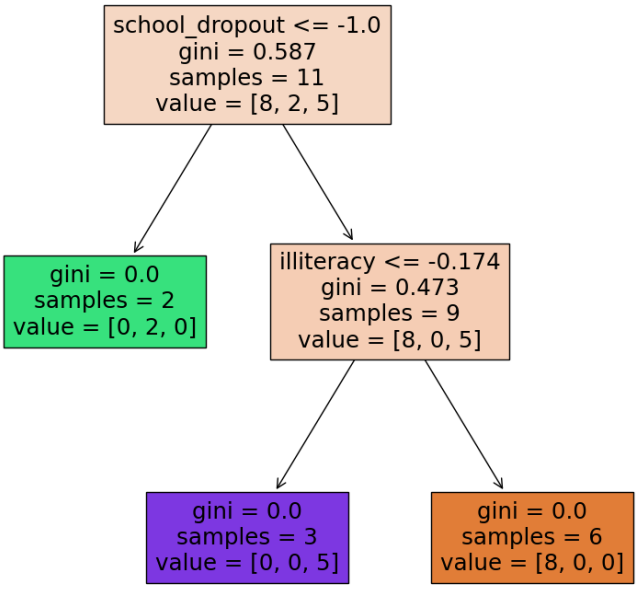
Por último, se vuelven a desprender por un lado, un nodo hoja con un índice Gini = 0, con muestras = 1 y valores de [3,0,0] representando 3 valores “High”. Por otro lado , el último nivel de decisión con la variable deserción escolar, presenta un índice Gini de 0.5, muestras = 2 y valores de [1,0,1].

De este último nivel se desprenden 2 nodos hojas, la primera presenta un índice Gini de 0, con una muestra, de valor “Low” [0,0,1]. Y el segundo nodo hoja, presenta un índice Gini de 0, con muestras = 1, con valor “High” [1,0,0].

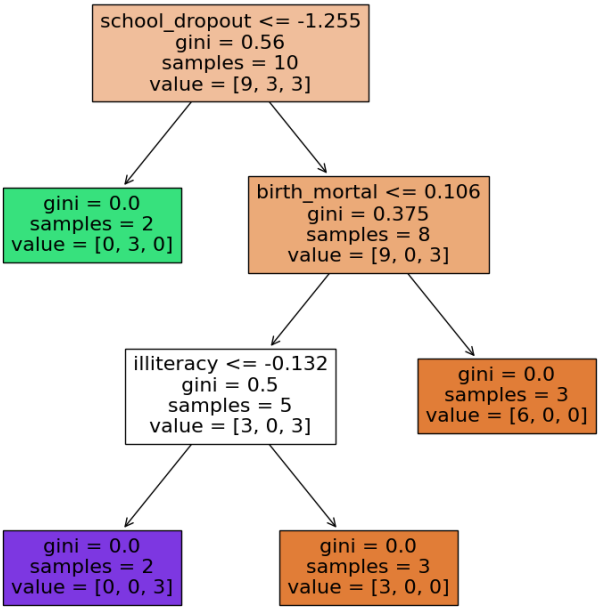
Para mejorar estos valores, decidimos ajustar los hiperparámetros del Random Forest Classifier, con un número de estimadores de 80 y profundidad del árbol de 4 niveles. Tomamos dos ejemplos, el primero con la posición 0 y el segundo árbol con la última posición 79.

<u>Con poda:</u>

Partición: 0.3
Estimadores: 80
Profundidad: 4
Posición: 0
Accuracy: 85.71



<u>Otro ej. con poda:</u>
Partición: 0.3
Estimadores: 80
Profundidad: 4
Posición: 79
Accuracy: 85.71



Realizando el análisis del árbol podado de la posición 79, pudimos visualizar que el nodo raíz refleja la variable deserción escolar, con un índice Gini del 0.56, un número de samples (muestras) de 10 y valores de [9,3,3] representando los niveles “High”, “Middle” y “Low” respectivamente.

Del nodo raíz se desprende una hoja y un nodo de decisión. El nodo de decisión (primer nivel) refleja la variable mortalidad infantil con un índice Gini de 0.375, un número de muestras de 8 y valores [9,0,3]. La hoja posee un Gini de 0, con 2 muestras y valores de [0,3,0], representando 3 valores “Middle” puros.

A su vez, de ese nodo decisión se desprende una hoja con un índice gini = 0, con 3 muestras y valores de [6,0,0], representando 6 valores “High” puros. También se desprende otro nodo de decisión que refleja la variable Analfabetismo (segundo nivel), con un índice Gini de 0.5, 5 muestras y valores de [3,0,3].

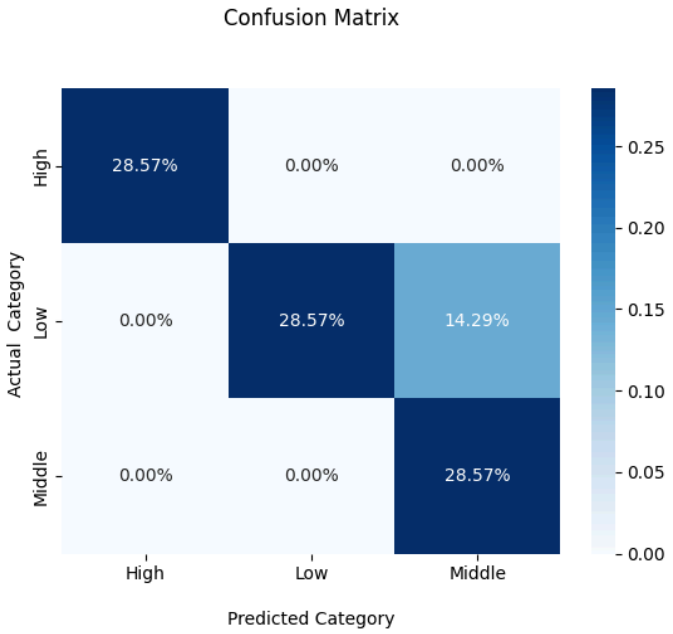
De este último nivel, se desprenden dos hojas, una con índice Gini 0, 3 muestras y valores de [3,0,0], representando 3 valores “High” puros, y la otra con un índice Gini de 0, 2 muestras y valores de [0,0,3], representando 3 valores “Low” puros.

Al evaluar los valores representados en este último árbol, se pudo determinar que los resultados fueron más eficientes y óptimos en comparación con el árbol sin podar.

Validación y Conclusión

Luego de una gran cantidad de evaluaciones realizadas y un análisis exhaustivo, se procedió a validar el modelo utilizando las métricas de Precision Score, Recall Score y Accuracy Score. Dicha validación se realizó tanto en el árbol sin podar como con el árbol podado, logrando obtener un modelo con un 85,71% de accuracy, con éste último.

Se importó y utilizó la Matriz de Confusión para evaluar y visualizar la performance de nuestro modelo.



Tomando ambos modelos de árboles de decisión, concluimos que el árbol podado dió mejores resultados de la variable respuesta.

El árbol podado posee un 85,71% de accuracy , mientras que el árbol sin podar posee un 71,43% de accuracy.

```
# Métrica 1: precision score
ps=precision_score(y_test,y_pred_model,average='macro')
round(ps*100,2)
```

```
88.89
```

```
# Métrica 2: recall score
rs=recall_score(y_test,y_pred_model,average='macro')
round(rs*100,2)
```

```
88.89
```

```
# Métrica 3: accuracy
accs=accuracy_score(y_test,y_pred_model)
round(accs*100,2)
```

```
85.71
```

```
# Resultados de las métricas
print("metric_1 (Precission_Score) :",round(ps*100,2))
print("metric_2 (Recall Score):",round(rs*100,2))
print("metric_3 (Accuracy):",round(accs*100,2))
```

```
metric_1 (Precission_Score) : 88.89
metric_2 (Recall Score): 88.89
metric_3 (Accuracy): 85.71
```