TRABAJO FINAL INTEGRADOR

El siguiente cuaderno corresponde al Trabajo Final Integrador del curso Python para Data Science dictado por la Academia Data School a través de la Mg. Ing. Layla Scheli

Consigna

Iniciaremos realizando un breve EDA (Análisis Exploratorio de Datos) del dataset:

- Aplicar los conocimientos adquiridos durante el curso para el análisis del dataset que se adjunta
- Importar las librerías necesarias para la realización del desafío (Ej: pandas, numpy, matplotlib, etc) y el dataset propuesto para la ejercitación.

Estructura del documento a entregar

1. Introducción: Descripción del dataset y sus características.
2. Desarrollo: Realizar un proceso exploratorio de datos. Incluir al menos 3 visualizaciones diferentes con sus correspondientes interpretaciones.
3. Conclusiones: De la actividad y de los insights obtenidos

```
#from google.colab import data_table
#data_table.enable_dataframe_formatter()
#data_table.disable_dataframe_formatter()
```

1. Introducción:

Importar las librerías necesarias para la realización del desafío (Ej: pandas, numpy, matplotlib, etc) y el dataset propuesto para la ejercitación.

```
# Librerias

import pandas as pd
import matplotlib.pyplot as plt
#%matplotlib inline
import seaborn as sns

# Dataset a importar

df = pd.read_excel(r'https://github.com/g-gutierr/educacion_sexual_TFI/blob/main/Educacion%20Sexual.xlsx?raw=true', engine="open
```

```
## Dimensiones del Dataset

df.shape
```

```
(15157, 6)
```

```
df.head(5)
```

	id	edad	anios_educ	en_pareja	num_hijos	bajo_socioecon
0	1	18	9	0	0	1
1	2	16	7	0	0	1
2	3	15	9	0	0	1
3	4	17	9	1	0	0
4	5	18	9	1	0	0

```
df.tail(5)
```

```
# Registros por columnas
df.dtypes
```

```
id               int64
edad             int64
anios_educ       int64
en_pareja        int64
num_hijos        int64
bajo_socioecon   int64
dtype: object
```

Tenemos que los features tienen variables del tipo entero

Las variables presentes son del tipo cuantitativas (edad, anios_educación, num_hijos) y categóricas(bajo_socioeconom, en_pareja)

```
# Descripción del dataset
df.describe()
```

	id	edad	anios_educ	en_pareja	num_hijos	bajo_socioecon
count	15157.00000	15157.000000	15157.000000	15157.000000	15157.000000	15157.000000
mean	7579.00000	16.962064	8.506763	0.161707	0.150887	0.250049
std	4375.59335	1.413214	1.176005	0.368194	0.409855	0.433056
min	1.00000	15.000000	6.000000	0.000000	0.000000	0.000000
25%	3790.00000	16.000000	8.000000	0.000000	0.000000	0.000000
50%	7579.00000	17.000000	9.000000	0.000000	0.000000	0.000000
75%	11368.00000	18.000000	9.000000	0.000000	0.000000	1.000000
max	15157.00000	19.000000	12.000000	1.000000	3.000000	1.000000

- A partir de la descripción se puede inferir que el count de cada feature tiene la misma cantidad que la cantidad de observaciones, por lo tanto no se tienen muestras con faltantes de datos
- A su vez se puede inferir de que el promedio de edad es de 17 años aproximadamente y que el promedio de educación es de 8 años y medio.
- Se puede observar que la edad varía entre los 15 y 19 años, el número de hijos entre 0 y 3, y los años de educación entre 6 y 12 años

```
## Se hace un DROP de la Columna ID ya que entiendo no aporta al análisis

df = df.drop(['id'], axis=1)
```

```
df.corr()
```

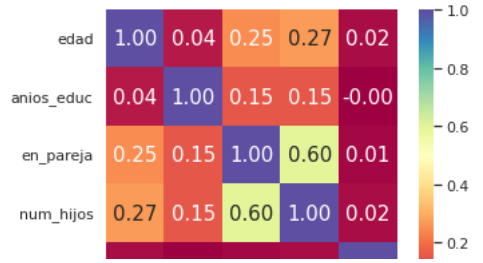
	edad	anios_educ	en_pareja	num_hijos	bajo_socioecon
edad	1.000000	0.040193	0.245869	0.270632	0.020460
anios_educ	0.040193	1.000000	0.154045	0.153597	-0.000989
en_pareja	0.245869	0.154045	1.000000	0.601265	0.013295
num_hijos	0.270632	0.153597	0.601265	1.000000	0.019753
bajo_socioecon	0.020460	-0.000989	0.013295	0.019753	1.000000

2. Desarrollo:

Realizar un proceso exploratorio de datos. Incluir al menos 3 visualizaciones diferentes con sus correspondientes interpretaciones.

```
# Se realiza un mapa de calor de la correlación de los parámetros
sns.set()
sns.heatmap(df.corr(),cbar = True, square = True, annot = True, fmt = '.2f', annot_kws = {'size': 15}, cmap = 'Spectral')
```

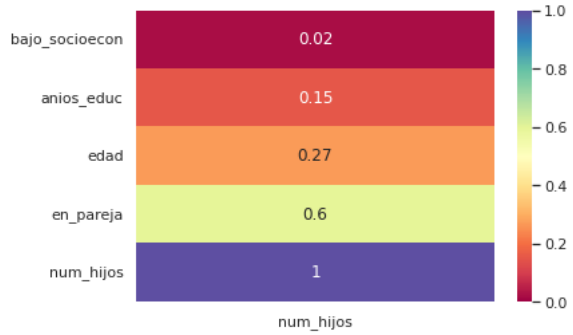
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f464388f670>
```



En esta visualizacion se observa que no hay features que muestren una correlación muy fuerte entre ellas. Se puede decir que la correlación mas grande se dá entre la variable 'en_pareja' y 'num_hijos'

```
sns.heatmap(df.corr()[['num_hijos']].sort_values(by='num_hijos', ascending=True), vmin=0, vmax=1, annot=True, cmap='Spectral')
```

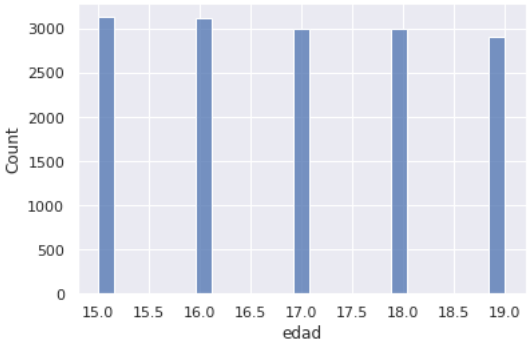
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4643b38370>
```



Visualización de correlación entre el número de hijos y las demás variables presentes en el dataset. Esta visualización se puede ver como un extracto de la visualización de correlaciones entre variables(df.corr). Toma a número de hijos como base y ordena las correlaciones de mayor a menor

```
## Veamos como están distribuidas las edades en el dataset
sns.histplot(data=df.edad)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4649405610>
```



En esta visualizacion del tipo histograma, podemos inferir que las edades del dataset tiene una distribucion similar a la distribución uniforme.

Parte ML

Una vez finalizado el EDA y encontradas aquellas variables que tienen mayor relación, se procede utilizar algoritmos de ML para determinar si existe un modelo que pueda predecir a partir de si está o no en pareja y la edad, los números de hijos

```
# Importacion de Librerias necesarias, principalmente sklearn y numpy

from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
import numpy as np
```

```
# Como se comento anteriormente, se mantienen features que tienen cierto grado de importancia en su correlacion

data = df[['edad','en_pareja','num_hijos']]
X=data.drop("num_hijos",axis=1)
y=data.num_hijos
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state = 42)
```

```
# Creacion de los modelos de regresion lineal, arbol de regresion y K Vecinos Cercanos

linear_model = LinearRegression()
tree_regressor = DecisionTreeRegressor(max_depth = 2, random_state = 42)
knn_regressor = KNeighborsRegressor(n_neighbors = 5)
```

```
tree_regressor.fit(X_train,y_train)
linear_model.fit(X_train,y_train)
knn_regressor.fit(X_train,y_train)
```

```
KNeighborsRegressor()
```

```
modelos = ['Regresion lineal','Árbol de Decisión', 'Vecinos más cercanos']
for i, model in enumerate([linear_model, tree_regressor, knn_regressor]):
 y_train_pred = model.predict(X_train)
 y_test_pred = model.predict(X_test)
 print(f'Modelo: {modelos[i]}')
 rmse_train = np.sqrt(mean_squared_error(y_train, y_train_pred))
 rmse_test = np.sqrt(mean_squared_error(y_test, y_test_pred))
 print(f'Raíz del error cuadrático medio en Train: {rmse_train}')
 print(f'Raíz del error cuadrático medio en Test: {rmse_test}')
```

```
Modelo: Regresion lineal
Raíz del error cuadrático medio en Train: 0.3203643104124595
Raíz del error cuadrático medio en Test: 0.33021571244028336
Modelo: Árbol de Decisión
Raíz del error cuadrático medio en Train: 0.31839432114903277
Raíz del error cuadrático medio en Test: 0.3301775949772468
Modelo: Vecinos más cercanos
Raíz del error cuadrático medio en Train: 0.3380537354081099
Raíz del error cuadrático medio en Test: 0.34672738119299634
```

En el caso donde se quiere estudiar estrictamente el embarazo adolescente, suponiendo que no es de interés la cantidad de hijos, sino si tiene hijos, podemos crear una variable auxiliar, con el fin de determinar si se puede determinar el embarazo adolescente.

```
df['tiene_hijos']=np.where(df['num_hijos']>0,1,0)
```

```
data2 = df[['edad','en_pareja','tiene_hijos']]
X=data2.drop("tiene_hijos",axis=1)
y=data2.tiene_hijos
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30, random_state = 42)
```

```
tree_regressor.fit(X_train,y_train)
linear_model.fit(X_train,y_train)
knn_regressor.fit(X_train,y_train)
```

```
KNeighborsRegressor()
```

```
modelos = ['Regresion lineal','Árbol de Decisión', 'Vecinos más cercanos']
for i, model in enumerate([linear_model, tree_regressor, knn_regressor]):
 y_train_pred = model.predict(X_train)
 y_test_pred = model.predict(X_test)
 print(f'Modelo: {modelos[i]}')
 rmse_train = np.sqrt(mean_squared_error(y_train, y_train_pred))
 rmse_test = np.sqrt(mean_squared_error(y_test, y_test_pred))
 print(f'Raíz del error cuadrático medio en Train: {rmse_train}')
 print(f'Raíz del error cuadrático medio en Test: {rmse_test}')
```

```
Modelo: Regresion lineal
Raíz del error cuadrático medio en Train: 0.26109345474350265
Raíz del error cuadrático medio en Test: 0.26074647501514203
Modelo: Árbol de Decisión
```

```
Raíz del error cuadrático medio en Train: 0.26012977009908733
Raíz del error cuadrático medio en Test: 0.2612035506442469
Modelo: Vecinos más cercanos
Raíz del error cuadrático medio en Train: 0.276138078984562
Raíz del error cuadrático medio en Test: 0.2767762654986148
```

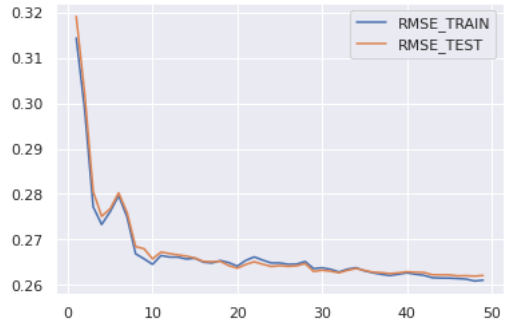
```
# Testeo de parámetros k
rmse_train = []
rmse_test = []
x_plot = []
for k in range(1,50,1):
  knn_regressor = KNeighborsRegressor(n_neighbors = k)
  knn_regressor.fit(X_train,y_train)

  modelos = ['Vecinos más cercanos']
  for i, model in enumerate([knn_regressor]):
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)
    x_plot.append(k)
    print(f'Modelo: {modelos[i]} iteracion: {k}')
    rmse_train.append(np.sqrt(mean_squared_error(y_train, y_train_pred)))
    rmse_test.append(np.sqrt(mean_squared_error(y_test, y_test_pred)))
#    print(f'Raíz del error cuadrático medio en Train: {rmse_train}')
#    print(f'Raíz del error cuadrático medio en Test: {rmse_test}')
```

```
Modelo: Vecinos más cercanos iteracion: 1
Modelo: Vecinos más cercanos iteracion: 2
Modelo: Vecinos más cercanos iteracion: 3
Modelo: Vecinos más cercanos iteracion: 4
Modelo: Vecinos más cercanos iteracion: 5
Modelo: Vecinos más cercanos iteracion: 6
Modelo: Vecinos más cercanos iteracion: 7
Modelo: Vecinos más cercanos iteracion: 8
Modelo: Vecinos más cercanos iteracion: 9
Modelo: Vecinos más cercanos iteracion: 10
Modelo: Vecinos más cercanos iteracion: 11
Modelo: Vecinos más cercanos iteracion: 12
Modelo: Vecinos más cercanos iteracion: 13
Modelo: Vecinos más cercanos iteracion: 14
Modelo: Vecinos más cercanos iteracion: 15
Modelo: Vecinos más cercanos iteracion: 16
Modelo: Vecinos más cercanos iteracion: 17
Modelo: Vecinos más cercanos iteracion: 18
Modelo: Vecinos más cercanos iteracion: 19
Modelo: Vecinos más cercanos iteracion: 20
Modelo: Vecinos más cercanos iteracion: 21
Modelo: Vecinos más cercanos iteracion: 22
Modelo: Vecinos más cercanos iteracion: 23
Modelo: Vecinos más cercanos iteracion: 24
Modelo: Vecinos más cercanos iteracion: 25
Modelo: Vecinos más cercanos iteracion: 26
Modelo: Vecinos más cercanos iteracion: 27
Modelo: Vecinos más cercanos iteracion: 28
Modelo: Vecinos más cercanos iteracion: 29
Modelo: Vecinos más cercanos iteracion: 30
Modelo: Vecinos más cercanos iteracion: 31
Modelo: Vecinos más cercanos iteracion: 32
Modelo: Vecinos más cercanos iteracion: 33
Modelo: Vecinos más cercanos iteracion: 34
Modelo: Vecinos más cercanos iteracion: 35
Modelo: Vecinos más cercanos iteracion: 36
Modelo: Vecinos más cercanos iteracion: 37
Modelo: Vecinos más cercanos iteracion: 38
Modelo: Vecinos más cercanos iteracion: 39
Modelo: Vecinos más cercanos iteracion: 40
Modelo: Vecinos más cercanos iteracion: 41
Modelo: Vecinos más cercanos iteracion: 42
Modelo: Vecinos más cercanos iteracion: 43
Modelo: Vecinos más cercanos iteracion: 44
Modelo: Vecinos más cercanos iteracion: 45
Modelo: Vecinos más cercanos iteracion: 46
Modelo: Vecinos más cercanos iteracion: 47
Modelo: Vecinos más cercanos iteracion: 48
Modelo: Vecinos más cercanos iteracion: 49
```

```
df1 = pd.DataFrame(list(zip(x_plot, rmse_train, rmse_test)))
df1.columns =['k', 'rmse_train', 'rmse_test']
```

```
## Plot del RMSE de Train y TEST
# plot lines
plt.plot(df1.k, df1.rmse_train, label = "RMSE_TRAIN")
plt.plot(df1.k, df1.rmse_test, label = "RMSE_TEST")
plt.legend()
plt.show()
```



Finalmente con esta visualizacion de linea se observa que para k mayor a 10 hay una tendencia a que el RMSE tanto de test como de train disminuye a un ritmo menor a que se da entre k=1 y k=8

3. Conclusiones:

A partir del EDA y principalmente con la función corr() de Pandas, nos permite ver cuales son los features que tienen importancia al momento de crear modelos de predicción de embarazo adolescente. En este caso las variables 'en_pareja' y 'edad' son las que mayor relación tenían.

Respecto a los features y a la creacion del feature tiene hijos, permite de mejor manera predecir embarazo adolescente.

El ajuste de parametros en el caso del ajuste del k para el algoritmo de knn tuvo una mejora en el RMSE.

Una visualizacion del tipo torta con porcentajes de las edades de las muestras hubiera sido de mayor utilidad al histograma ya que de esa manera se da directamente una mejor idea de que tan uniforme es la distribucion.

En este caso donde tenemos variables discretas, scatterplots o pairplots no fueron de mucha utilidad, ya que no muestran claramente correlaciones entre features.