Importation des Libraires

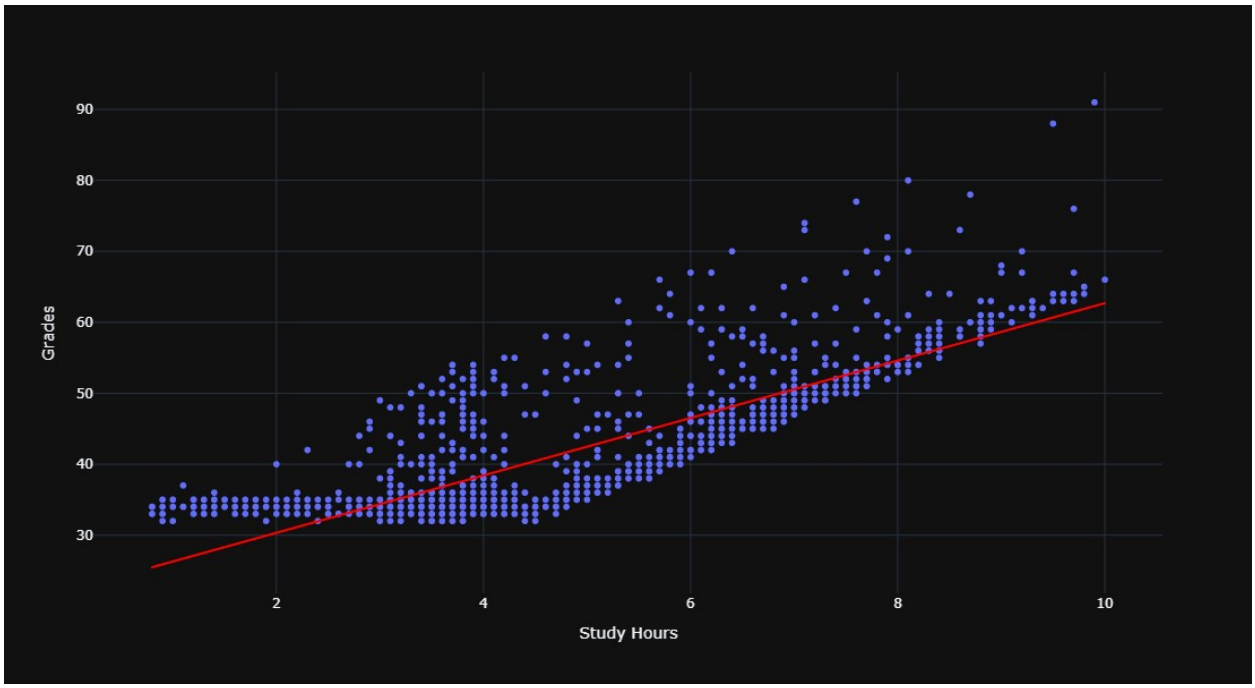
```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go

df=pd.read_csv("data.csv")
df
```

```
      Socioeconomic Score  Study Hours  Sleep Hours  Attendance (%)
Grades
0                 0.95822          3.4          8.2            53.0
47.0
1                 0.85566          3.2          5.9            55.0
35.0
2                 0.68025          3.2          9.3            41.0
32.0
3                 0.25936          3.2          8.2            47.0
34.0
4                 0.60447          3.8         10.0            75.0
33.0
...                   ...          ...          ...             ...
...
1383              0.44549          5.5          8.0            51.0
41.0
1384              0.52466          4.9          6.5            63.0
37.0
1385              0.88197          3.9          6.2            54.0
36.0
1386              0.47336          3.5          7.3            61.0
34.0
1387              0.58119          3.7          9.7            79.0
35.0

[1388 rows x 5 columns]
```

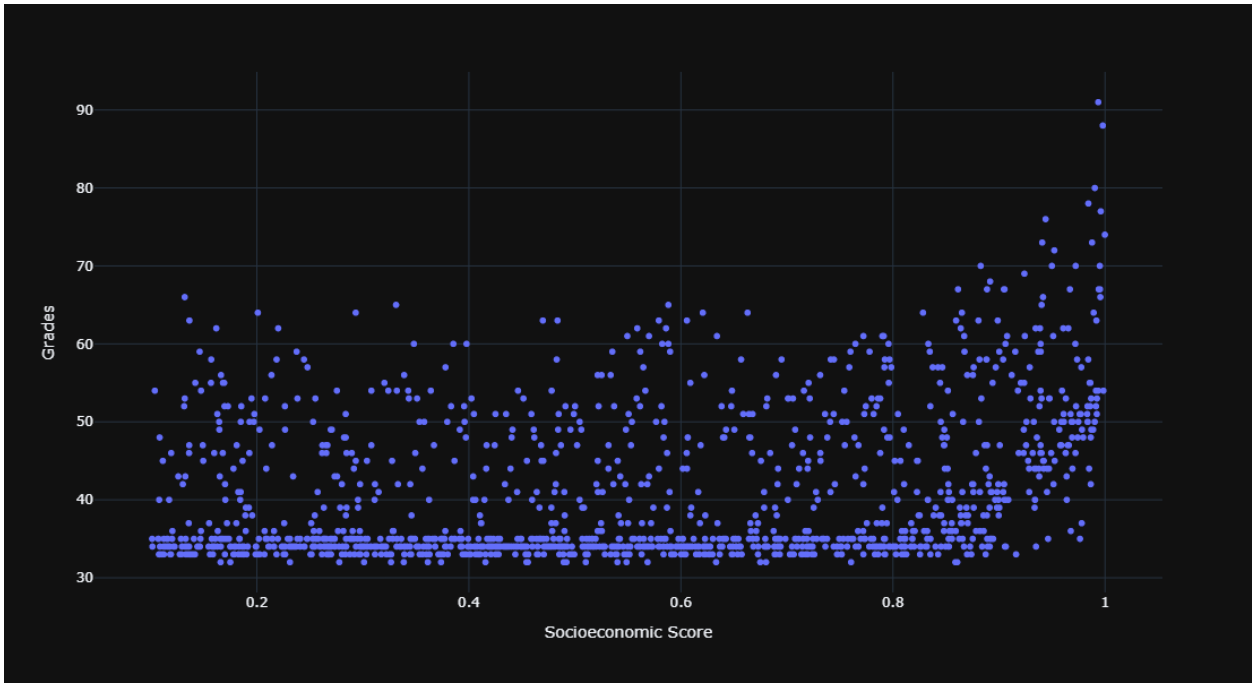
```
df["Grades"].max()
```

```
91.0
```

```
fig = px.scatter(df,x="Study
Hours",y="Grades",template="plotly_dark",width=1100,
height=600,trendline="ols",trendline_color_override="Red")
fig.show()
```




```
fig = px.scatter(df,x="Socioeconomic Score",y="Grades",template="plotly_dark",width=1100, height=600)
fig.show()
```




```
df
```

```
      Socioeconomic Score  Study Hours  Sleep Hours  Attendance (%)  Grades
```

```
0                  0.95822          3.4           8.2             53.0
47.0
1                  0.85566          3.2           5.9             55.0
35.0
2                  0.68025          3.2           9.3             41.0
32.0
3                  0.25936          3.2           8.2             47.0
34.0
4                  0.60447          3.8          10.0             75.0
33.0
...                    ...          ...           ...              ...
...
1383               0.44549          5.5           8.0             51.0
41.0
1384               0.52466          4.9           6.5             63.0
37.0
1385               0.88197          3.9           6.2             54.0
36.0
1386               0.47336          3.5           7.3             61.0
34.0
1387               0.58119          3.7           9.7             79.0
35.0

[1388 rows x 5 columns]
```

```
x=df.drop("Grades",axis=1)
y=df["Grades"]
```

```
x
```

```
      Socioeconomic Score  Study Hours  Sleep Hours  Attendance (%)
0                 0.95822          3.4           8.2             53.0
1                 0.85566          3.2           5.9             55.0
2                 0.68025          3.2           9.3             41.0
3                 0.25936          3.2           8.2             47.0
4                 0.60447          3.8          10.0             75.0
...                   ...          ...           ...              ...
1383              0.44549          5.5           8.0             51.0
1384              0.52466          4.9           6.5             63.0
1385              0.88197          3.9           6.2             54.0
1386              0.47336          3.5           7.3             61.0
1387              0.58119          3.7           9.7             79.0

[1388 rows x 4 columns]
```

```
y
```

```
0       47.0
1       35.0
2       32.0
3       34.0
```

```
4       33.0
        ...
1383    41.0
1384    37.0
1385    36.0
1386    34.0
1387    35.0
Name: Grades, Length: 1388, dtype: float64
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size=0.2,random_state=12)

reg = LinearRegression()

reg.fit(x_train,y_train)
```

```
LinearRegression()
```

```
reg.score(x_test,y_test)
```

```
0.7798040499606899
```

```
[[y]]
```

```
[[0       47.0
  1       35.0
  2       32.0
  3       34.0
  4       33.0
          ...
  1383    41.0
  1384    37.0
  1385    36.0
  1386    34.0
  1387    35.0
  Name: Grades, Length: 1388, dtype: float64]]
```

```
y_predict = reg.predict(x)
y_predict
```

```
array([40.88678721, 38.4778622 , 37.64834125, ..., 41.88994702,
       34.87208279, 35.92167128])
```

```
regression = pd.DataFrame({"Valeurs Réelles":y,"Valeurs
Prédites":y_predict})
regression
```

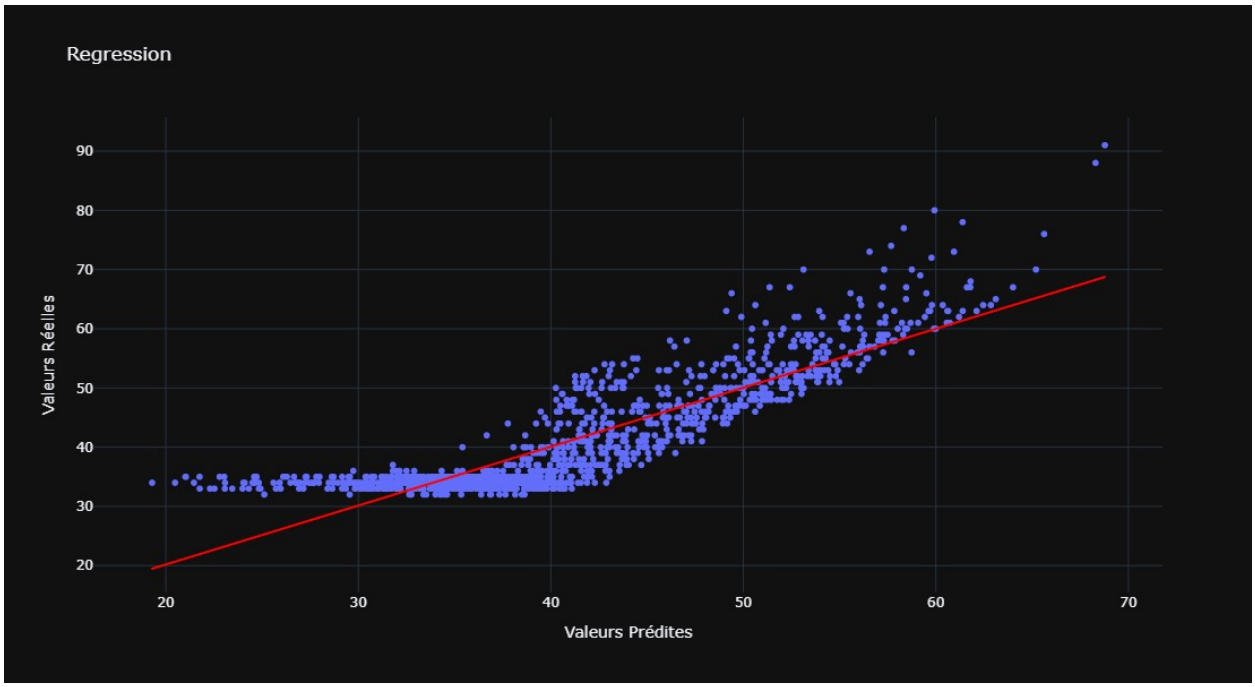
```
      Valeurs Réelles  Valeurs Prédites
0                47.0         40.886787
1                35.0         38.477862
```

```
2                32.0          37.648341
3                34.0          32.097328
4                33.0          36.938745
...               ...                ...
1383             41.0          43.886596
1384             37.0          41.284136
1385             36.0          41.889947
1386             34.0          34.872083
1387             35.0          35.921671

[1388 rows x 2 columns]
```

```
regression["Valeurs Prédites"].max()
```

```
68.7755176434648
```

```
fig=px.scatter(regression,x="Valeurs Prédites",y="Valeurs
Réelles",template="plotly_dark",title="Regression",width=1100,height=6
00,trendline="ols",trendline_color_override="red")
fig.show()
```



Essayons avec la classification (KNN)

Pour ce cas (KNN) on va travailler avec les valeurs 1(avoir la moyenne) et 0 (Ne pas avoir la moyenne)

```
df
```

```
      Socioeconomic Score  Study Hours  Sleep Hours  Attendance (%)
Grades
0                 0.95822          3.4          8.2            53.0
47.0
1                 0.85566          3.2          5.9            55.0
35.0
2                 0.68025          3.2          9.3            41.0
32.0
3                 0.25936          3.2          8.2            47.0
34.0
4                 0.60447          3.8         10.0            75.0
33.0
...                   ...          ...          ...             ...
...
1383              0.44549          5.5          8.0            51.0
41.0
1384              0.52466          4.9          6.5            63.0
37.0
1385              0.88197          3.9          6.2            54.0
36.0
1386              0.47336          3.5          7.3            61.0
34.0
1387              0.58119          3.7          9.7            79.0
35.0

[1388 rows x 5 columns]
```

```
df['Grades'] = df['Grades'].apply(lambda x: 1 if x > 50 else 0)
df
```

```
      Socioeconomic Score  Study Hours  Sleep Hours  Attendance (%)
Grades
0                 0.95822          3.4          8.2            53.0
0
1                 0.85566          3.2          5.9            55.0
0
2                 0.68025          3.2          9.3            41.0
0
3                 0.25936          3.2          8.2            47.0
0
4                 0.60447          3.8         10.0            75.0
0
...                   ...          ...          ...             ...
...
1383              0.44549          5.5          8.0            51.0
0
1384              0.52466          4.9          6.5            63.0
0
1385              0.88197          3.9          6.2            54.0
0
```

```
1386              0.47336          3.5           7.3              61.0
0
1387              0.58119          3.7           9.7              79.0
0

[1388 rows x 5 columns]
```

```
(df['Grades']==1).sum()
```

```
241
```

```
(df['Grades']==0).sum()
```

```
1147
```

```
x=df.drop(["Grades"],axis=1)
y=df["Grades"]

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size=0.2,random_state=0)

algo=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)

algo.fit(x_train,y_train)
```

```
KNeighborsClassifier()
```

```
y_pred = algo.predict(x_test)
```

Pour connaitre le score de notre algo

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# Calculer la précision
accuracy = accuracy_score(y_test, y_pred)
print(f"Précision du modèle : {accuracy:.2f}")
```

```
Précision du modèle : 0.92
```

Un score 0,92 !!! C'est très bien

Maintenant on va voir le cas de quelques elèves

```
df.columns = ["Socioeconomic_Score", "Study_Hours", "Sleep_Hours",
"Attendance","Grades"]
```

```
def verdict
(algo,Socioeconomic_Score,Study_Hours,Sleep_Hours,Attendance) :

x=np.array([Socioeconomic_Score,Study_Hours,Sleep_Hours,Attendance]).r
eshape(1,4)
    if algo.predict(x)==[1]:
        print('Réussi')
    else:
        print('Echec')


# Cas 1
verdict(algo,7,10,8,50)
```

```
Réussi

C:\Anaconda\Lib\site-packages\sklearn\base.py:493: UserWarning:

X does not have valid feature names, but KNeighborsClassifier was
fitted with feature names
```

```
# Cas 2
verdict(algo,4,10,2,60)
```

```
Réussi

C:\Anaconda\Lib\site-packages\sklearn\base.py:493: UserWarning:

X does not have valid feature names, but KNeighborsClassifier was
fitted with feature names
```

```
#Cas 3
verdict(algo,5,3,3,50)
```

```
Echec

C:\Anaconda\Lib\site-packages\sklearn\base.py:493: UserWarning:

X does not have valid feature names, but KNeighborsClassifier was
fitted with feature names
```

On calcule mâintenant la probabilité

```
probabilities = algo.predict_proba(x_test)

probabilities
```

```
array([[0.6, 0.4],
       [1. , 0. ],
```

```
       [0.6, 0.4],
       [0.8, 0.2],
       [0. , 1. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [0. , 1. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.6, 0.4],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.4, 0.6],
       [0.8, 0.2],
       [1. , 0. ],
       [0.6, 0.4],
       [1. , 0. ],
       [0.6, 0.4],
       [0.4, 0.6],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [0.4, 0.6],
       [0.8, 0.2],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.2, 0.8],
       [0.4, 0.6],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
```

```
       [1. , 0. ],
       [0. , 1. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.4, 0.6],
       [1. , 0. ],
       [1. , 0. ],
       [0. , 1. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [0.4, 0.6],
       [0. , 1. ],
       [0.8, 0.2],
       [0.8, 0.2],
       [0.8, 0.2],
       [0.6, 0.4],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0. , 1. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [0.6, 0.4],
       [0.8, 0.2],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [0. , 1. ],
       [1. , 0. ],
       [1. , 0. ],
```

```
       [0.8, 0.2],
       [0.6, 0.4],
       [0.4, 0.6],
       [1. , 0. ],
       [0.6, 0.4],
       [1. , 0. ],
       [1. , 0. ],
       [0.2, 0.8],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [0.2, 0.8],
       [0.2, 0.8],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [0.4, 0.6],
       [1. , 0. ],
       [1. , 0. ],
       [0. , 1. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [0.8, 0.2],
       [1. , 0. ],
```

```
       [0.6, 0.4],
       [0.8, 0.2],
       [0. , 1. ],
       [0.4, 0.6],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.6, 0.4],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.6, 0.4],
       [0. , 1. ],
       [0.8, 0.2],
       [0.4, 0.6],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.2, 0.8],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.2, 0.8],
       [1. , 0. ],
       [0. , 1. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.6, 0.4],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
```

```
       [0.8, 0.2],
       [0.8, 0.2],
       [0. , 1. ],
       [1. , 0. ],
       [0.6, 0.4],
       [0.4, 0.6],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [0.2, 0.8],
       [1. , 0. ],
       [1. , 0. ],
       [0.6, 0.4],
       [1. , 0. ],
       [1. , 0. ],
       [0.2, 0.8],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.4, 0.6],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.2, 0.8],
       [0.8, 0.2],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.2, 0.8],
       [1. , 0. ],
       [1. , 0. ],
       [0.4, 0.6],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.6, 0.4],
       [0.8, 0.2],
```

```
       [0.8, 0.2],
       [0.4, 0.6],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.6, 0.4],
       [1. , 0. ],
       [1. , 0. ],
       [0. , 1. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.4, 0.6],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ]])
```

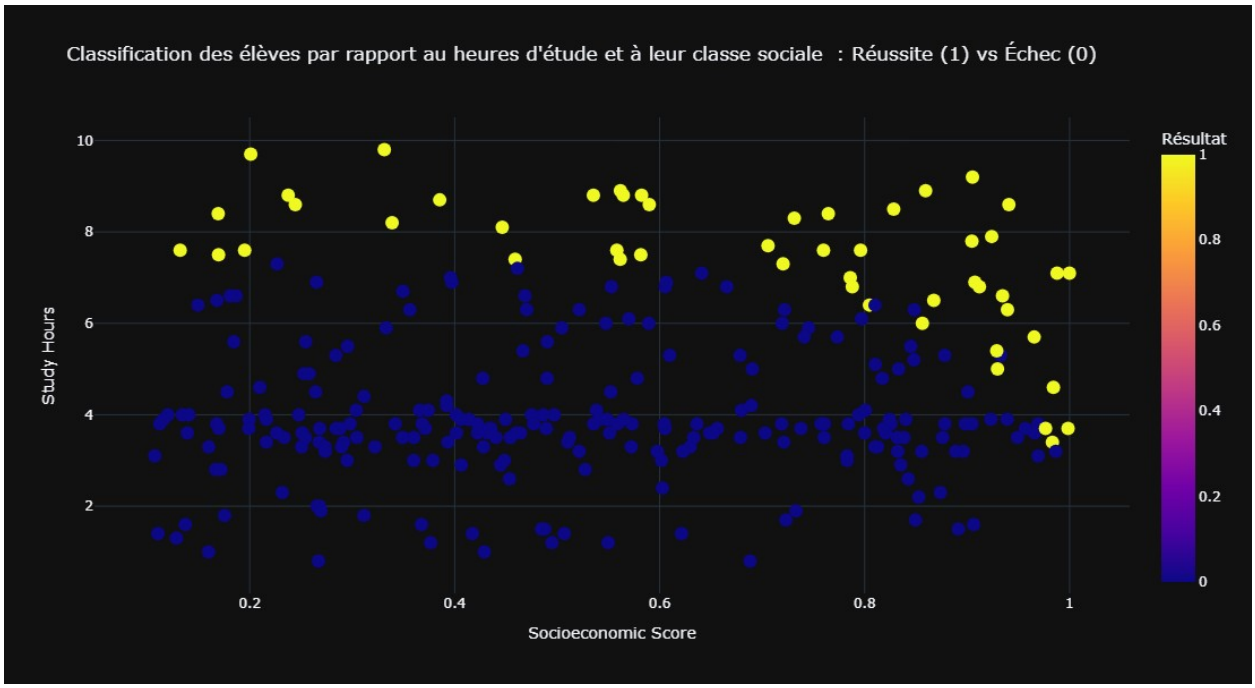
```
def proba(algo,Socioeconomic_Score,Study_Hours,Sleep_Hours,Attendance)
:

a=np.array([Socioeconomic_Score,Study_Hours,Sleep_Hours,Attendance]).r
eshape(1,4)
    proba=algo.predict_proba(a)
    print(f"{proba}")

proba(algo,5,3,3,50)
```

```
[[1. 0.]]
```

```
C:\Anaconda\Lib\site-packages\sklearn\base.py:493: UserWarning:

X does not have valid feature names, but KNeighborsClassifier was
fitted with feature names
```

```
proba(algo,3,5,3,50)
```

```
[[1. 0.]]

C:\Anaconda\Lib\site-packages\sklearn\base.py:493: UserWarning:

X does not have valid feature names, but KNeighborsClassifier was
fitted with feature names


# Exemple : Scatter plot pour visualiser ceux qui ont réussi vs échoué
fig = px.scatter(
    x_test,
    x="Socioeconomic Score",  # Axe X
        y="Study Hours",   # Axe Y
    color=y_test,

                # Coloration selon le résultat (succès/échec)
    labels={'color': 'Résultat'},  # Légende
    title="Classification des élèves par rapport au heures d'étude et
à leur classe sociale  : Réussite (1) vs Échec (0)",
    template="plotly_dark",
    width=1100,
    height=600,

)

# Mise en forme du graphique
fig.update_traces(marker_size=12)  # Ajuste la taille des points
fig.update_layout(legend_orientation='h')  # Légende horizontale

fig.show()
```



Maintenant on va Polt les probabilités

```
x_test
```

	Socioeconomic Score	Study Hours	Sleep Hours	Attendance (%)
667	0.45895	7.4	9.4	41.0
312	0.18435	5.6	5.2	52.0
1030	0.52157	6.3	8.7	65.0
813	0.87821	5.3	5.1	48.0
141	0.53531	8.8	6.7	61.0
...	...	...	...	...
1333	0.44637	8.1	8.5	72.0
467	0.32222	3.3	8.8	53.0
936	0.37412	4.1	8.0	65.0
618	0.42228	3.8	8.2	61.0
1316	0.55228	4.5	7.0	54.0

```
[278 rows x 4 columns]
```

```
df_plot = x_test.copy()
```

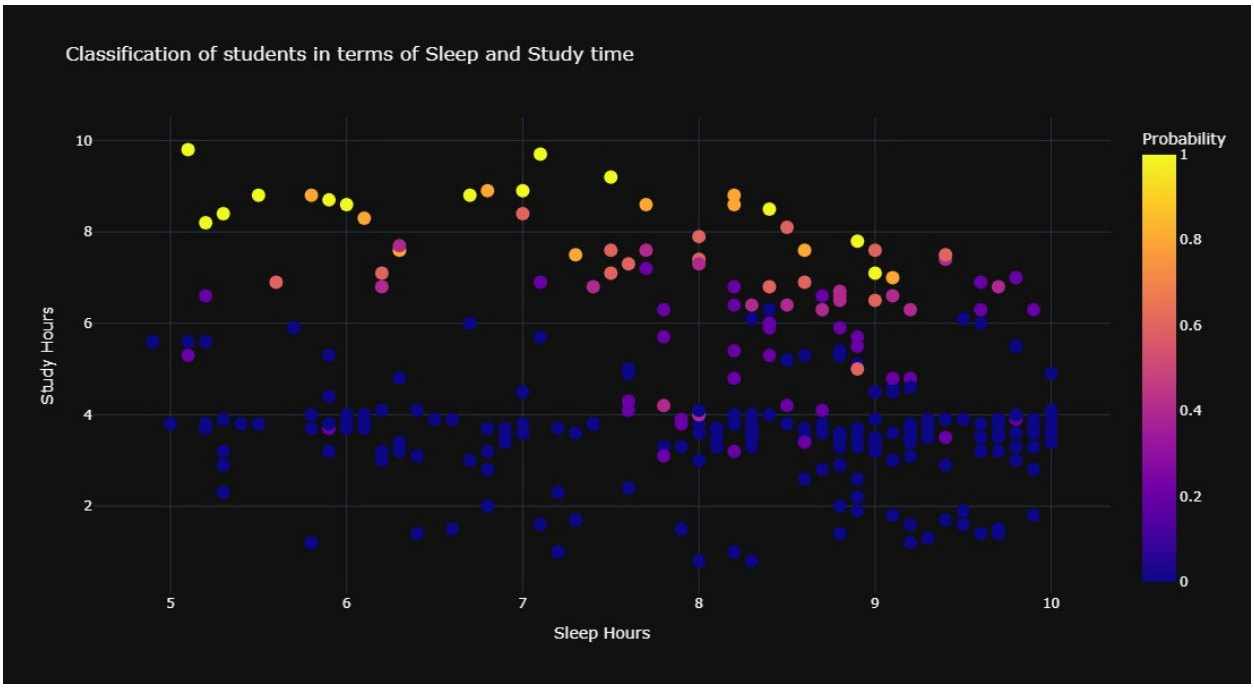
```
df_plot["Proba"]=probabilities[:,1]
```

```
df_plot
```

	Socioeconomic Score	Study Hours	Sleep Hours	Attendance (%)
Proba				
667	0.45895	7.4	9.4	41.0
0.4				

```
312                0.18435          5.6          5.2                52.0
0.0
1030               0.52157          6.3          8.7                65.0
0.4
813                0.87821          5.3          5.1                48.0
0.2
141                0.53531          8.8          6.7                61.0
1.0
...                    ...          ...          ...                 ...
...
1333               0.44637          8.1          8.5                72.0
0.6
467                0.32222          3.3          8.8                53.0
0.0
936                0.37412          4.1          8.0                65.0
0.0
618                0.42228          3.8          8.2                61.0
0.0
1316               0.55228          4.5          7.0                54.0
0.0

[278 rows x 5 columns]
```

```
fig = px.scatter(
    df_plot,
    x="Sleep Hours",  # Axe X
        y="Study Hours",   # Axe Y
    color=probabilities[:,1],
                # Coloration selon le résultat (succès/échec)
    labels={'color': 'Probability'},  # Légende
    title="Classification of students in terms of Sleep and Study
time",
    template="plotly_dark",
    width=1100,
    height=600,

)

# Mise en forme du graphique
fig.update_traces(marker_size=12)  # Ajuste la taille des points
fig.update_layout(legend_orientation='h')  # Légende horizontale

fig.show()
```



Cross Validation

```
from sklearn.model_selection import cross_val_score

cross_val_score(algo,x,y).mean()

0.907074773394281
```