Comece a programar ou gere código com IA.

*EXEMPLO *

objetivo

Tentar identificar correlações da performace do estudo online utilizando algoritmos de classificação em ML

*Dados de exemplo de um Dataset retirado do Kaggle

*Dados não minerados

```
#Bibliotecas default
import pandas as pd
import seaborn as sns
#Importação de classificadores
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
```

```
/usr/local/lib/python3.10/dist-packages/dask/dataframe/__init__.py:42: FutureWarning:
Dask dataframe query planning is disabled because dask-expr is not installed.

You can install it with `pip install dask[dataframe]` or `conda install dask`.
This will raise in a future version.

  warnings.warn(msg, FutureWarning)
```

```
#Install do dask[DataFrame]
pip install dask[dataframe]
```

```
Requirement already satisfied: dask[dataframe] in /usr/local/lib/python3.10/dist-packages (2024.10.0)
Requirement already satisfied: click>=8.1 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]) (8.1.7)
Requirement already satisfied: cloudpickle>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]) (3.1.0)
Requirement already satisfied: fsspec>=2021.09.0 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]) (2024.10.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]) (24.2)
Requirement already satisfied: partd>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]) (1.4.2)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]) (6.0.2)
Requirement already satisfied: toolz>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]) (0.12.1)
Requirement already satisfied: importlib-metadata>=4.13.0 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]) (8.5.0)
Requirement already satisfied: pandas>=2.0 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]) (2.2.2)
Collecting dask-expr<1.2,>=1.1 (from dask[dataframe])
  Downloading dask_expr-1.1.21-py3-none-any.whl.metadata (2.6 kB)
INFO: pip is looking at multiple versions of dask-expr to determine which version is compatible with other requirements. This could
  Downloading dask_expr-1.1.20-py3-none-any.whl.metadata (2.6 kB)
  Downloading dask_expr-1.1.19-py3-none-any.whl.metadata (2.6 kB)
  Downloading dask_expr-1.1.18-py3-none-any.whl.metadata (2.6 kB)
  Downloading dask_expr-1.1.16-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: pyarrow>=14.0.1 in /usr/local/lib/python3.10/dist-packages (from dask-expr<1.2,>=1.1->dask[dataframe
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.10/dist-packages (from importlib-metadata>=4.13.0->dask[datafran
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0->dask[dataframe]) (1.26.4
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0->dask[dataframe]
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0->dask[dataframe]) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0->dask[dataframe]) (2024.2
Requirement already satisfied: locket in /usr/local/lib/python3.10/dist-packages (from partd>=1.4.0->dask[dataframe]) (1.0.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=2.0->dask[d
Downloading dask_expr-1.1.16-py3-none-any.whl (243 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 243.2/243.2 kB 6.4 MB/s eta 0:00:00
Installing collected packages: dask-expr
Successfully installed dask-expr-1.1.16
```

```
import os
# Listar arquivos no diretório atual
print(os.listdir('/content/'))
```

```
['.config', 'Math.csv', '.ipynb_checkpoints', 'sample_data']
```

```
data= pd.read_csv('/content/Math.csv')
```

```
#Verificação das primeiras linhas
print(data.head())
```

```
  Gender Home Location Level of Education  Age(Years)  Number of Subjects  \
0   Male         Urban     Under Graduate          18                  11
1   Male         Urban     Under Graduate          19                   7
2   Male         Rural     Under Graduate          18                   5
3   Male         Urban     Under Graduate          18                   5
4   Male         Rural     Under Graduate          18                   5

  Device type used to attend classes Economic status  Family size  \
0                             Laptop    Middle Class            4
1                             Laptop    Middle Class            4
2                             Laptop    Middle Class            5
3                             Laptop    Middle Class            4
4                             Laptop    Middle Class            4

   Internet facility in your locality Are you involved in any sports?  ...  \
0                                   5                              No  ...
1                                   1                             Yes  ...
2                                   2                              No  ...
3                                   4                             Yes  ...
4                                   3                              No  ...

  Time spent on social media (Hours)  Interested in Gaming?  \
0                                  1                     No
1                                  1                    Yes
2                                  1                     No
3                                  2                     No
4                                  2                    Yes

   Have separate room for studying?  Engaged in group studies?  \
0                                No                         No
1                               Yes                         No
2                               Yes                         No
3                                No                        yes
4                               Yes                        yes

  Average marks scored before pandemic in traditional classroom  \
0                                             91-100
1                                             91-100
2                                              71-80
3                                             91-100
4                                              81-90

  Your interaction in online mode  \
0                               1
1                               1
2                               1
3                               1
4                               3

  Clearing doubts with faculties in online mode Interested in?  \
0                                             1      Practical
1                                             1         Theory
2                                             1           Both
3                                             2         Theory
4                                             3           Both

   Performance in online  Your level of satisfaction in Online Education
0                      6                                         Average
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1033 entries, 0 to 1032
Data columns (total 23 columns):
 #   Column                                                         Non-Null Count  Dtype
---  ------                                                         --------------  -----
 0   Gender                                                         1033 non-null   object
 1   Home Location                                                  1033 non-null   object
 2   Level of Education                                             1033 non-null   object
 3   Age(Years)                                                     1033 non-null   int64
 4   Number of Subjects                                             1033 non-null   int64
 5   Device type used to attend classes                             1033 non-null   object
 6   Economic status                                                1033 non-null   object
 7   Family size                                                    1033 non-null   int64
 8   Internet facility in your locality                             1033 non-null   int64
 9   Are you involved in any sports?                                1033 non-null   object
 10  Do elderly people monitor you?                                 1033 non-null   object
 11  Study time (Hours)                                             1033 non-null   int64
 12  Sleep time (Hours)                                             1033 non-null   int64
 13  Time spent on social media (Hours)                             1033 non-null   int64
 14  Interested in Gaming?                                          1033 non-null   object
 15  Have separate room for studying?                               1033 non-null   object
 16  Engaged in group studies?                                      1033 non-null   object
 17  Average marks scored before pandemic in traditional classroom  1033 non-null   object
 18  Your interaction in online mode                                1033 non-null   int64
 19  Clearing doubts with faculties in online mode                  1033 non-null   int64
```

```
 20  Interested in?                                          1033 non-null   object
 21  Performance in online                                   1033 non-null   int64
 22  Your level of satisfaction in Online Education          1033 non-null   object
dtypes: int64(10), object(13)
memory usage: 185.7+ KB
```

```
# Verificando os valores únicos de cada coluna do tipo object
for column in data.select_dtypes(include=['object']).columns:
    print(f"{column}: {data[column].unique()}")
```

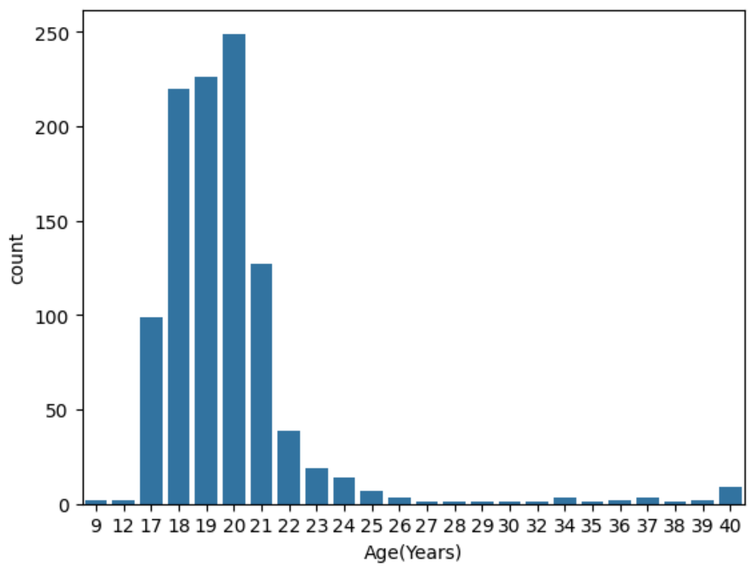
```
Gender: ['Male' 'Female']
Home Location: ['Urban' 'Rural']
Level of Education: ['Under Graduate' 'Post Graduate' 'School']
Device type used to attend classes: ['Laptop' 'Desktop' 'Mobile']
Economic status: ['Middle Class' 'Poor' 'Rich']
Are you involved in any sports?: ['No' 'Yes']
Do elderly people monitor you?: ['Yes' 'No']
Interested in Gaming?: ['No' 'Yes']
Have separate room for studying?: ['No' 'Yes']
Engaged in group studies?: ['No' 'yes']
Average marks scored before pandemic in traditional classroom: ['91-100' '71-80' '81-90' '61-70' '31-40' '41-50' '21-30' '11-20' '5:
 '0-10']
Interested in?: ['Practical' 'Theory' 'Both']
Your level of satisfaction in Online Education: ['Average' 'Bad' 'Good']
```

```
from sklearn.preprocessing import LabelEncoder
```

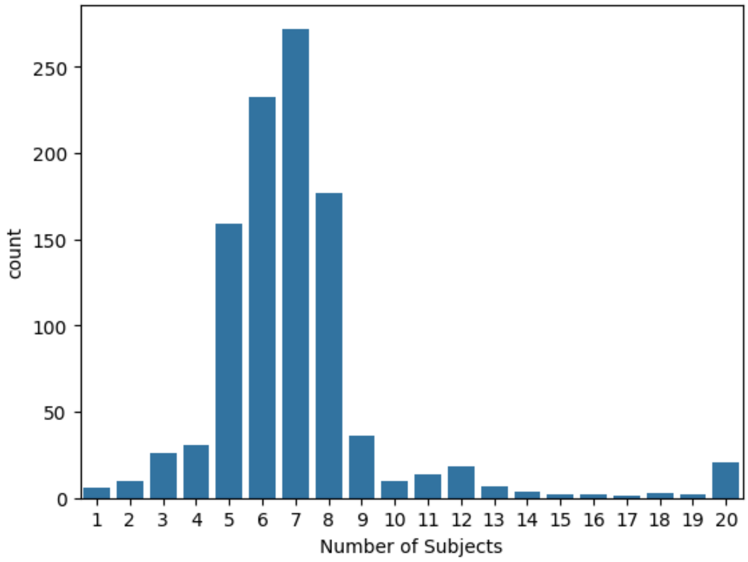
```
# Verificando os valores únicos de cada coluna do tipo inteiro
for column in data.select_dtypes(include=['int64']).columns:
    print(f"{column}: {data[column].unique()}")
```

```
Age(Years): [18 19 17 20 25 21 23 24 22 26  9 38 37 12 40 34 27 28 30 32 39 35 29 36]
Number of Subjects: [11  7  5  4  9  6 20  8  3  2 17 15  1 14 16 18 12 10 19 13]
Family size: [ 4  5  3  2  6  7  9 10  8]
Internet facility in your locality: [5 1 2 4 3]
Study time (Hours): [ 3  7  6  8  2  4  5  1 10  9]
Sleep time (Hours): [ 6  5  7  8  9  2 10  3  4  1]
Time spent on social media (Hours): [ 1  2  3  6  5  4  8 10  7  9]
Your interaction in online mode: [1 3 4 2 5]
Clearing doubts with faculties in online mode: [1 2 3 4 5]
Performance in online: [ 6  3  4  2  9  7  5 10  8]
```

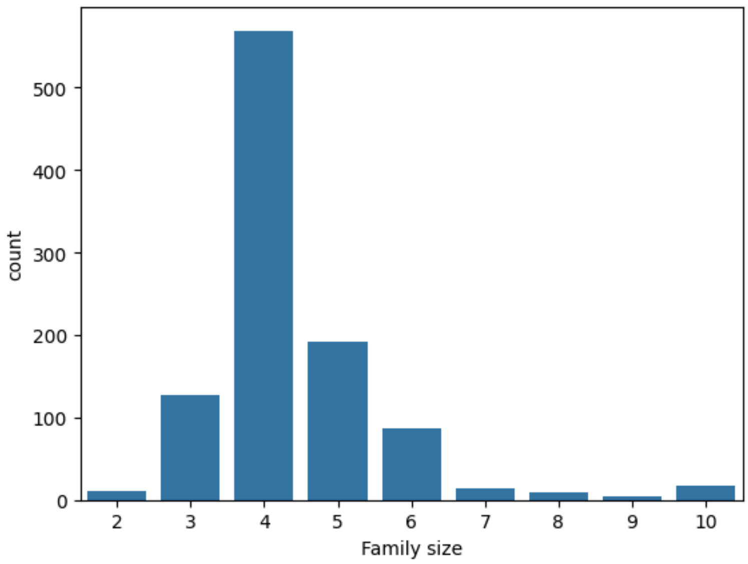
```
#Analise exploratória da variável 0
data['Age(Years)'].value_counts()
graf = sns.countplot(x ='Age(Years)', data=data)
```



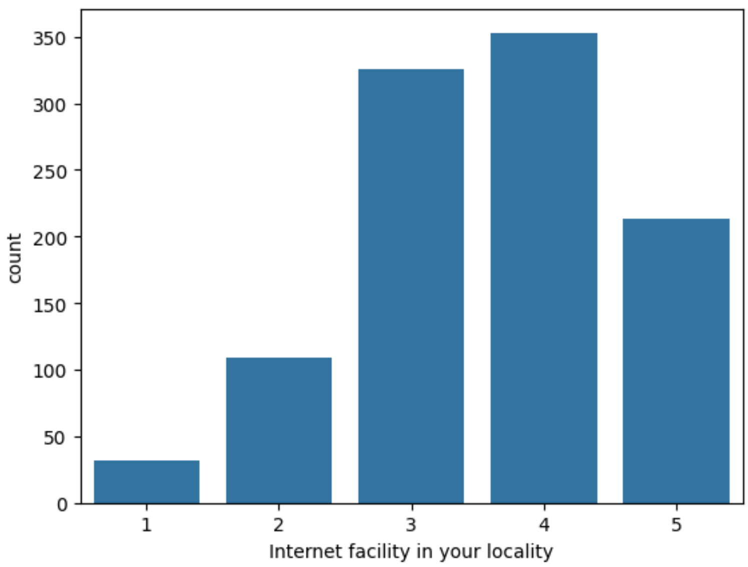

```
#Analise exploratória da variável 1
data['Number of Subjects'].value_counts()
graf = sns.countplot(x ='Number of Subjects', data=data)
```



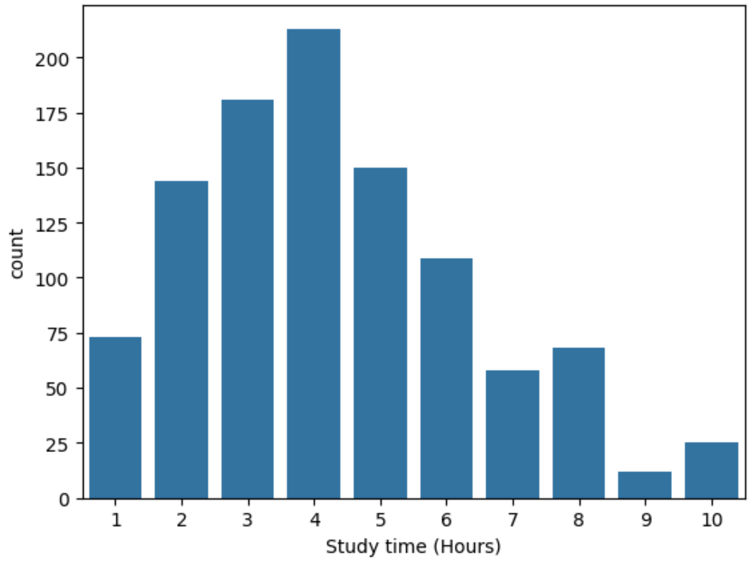

```
#Analise exploratória da variável 2
data['Family size'].value_counts()
graf = sns.countplot(x ='Family size', data=data)
```



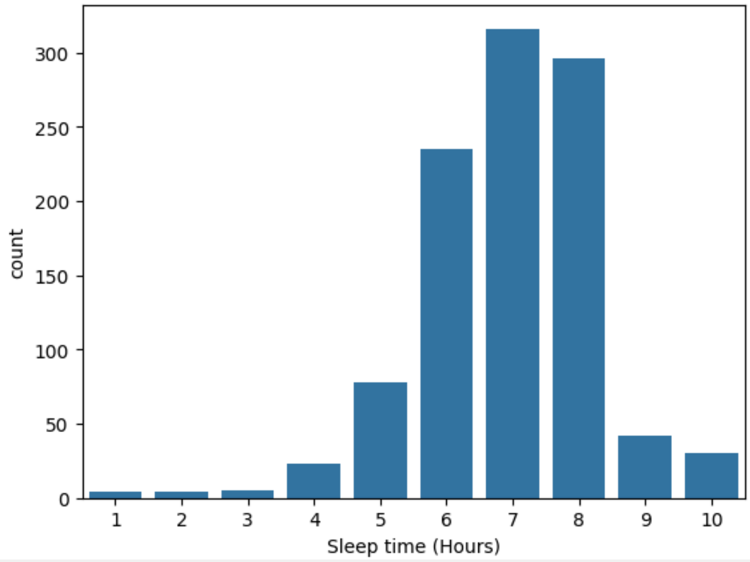

```
#Analise exploratória da variável 3
data['Internet facility in your locality'].value_counts()
graf = sns.countplot(x ='Internet facility in your locality', data=data)
```



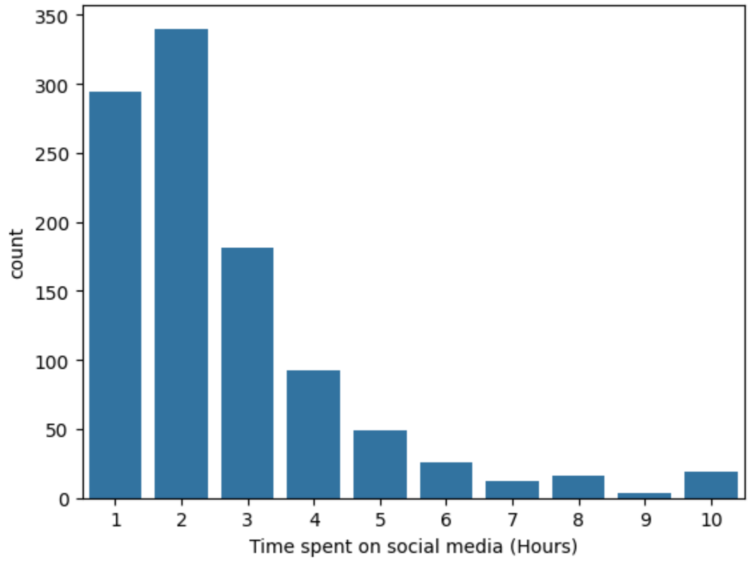
```
#Analise exploratória da variável 4
data['Study time (Hours)'].value_counts()
graf = sns.countplot(x ='Study time (Hours)', data=data)
```



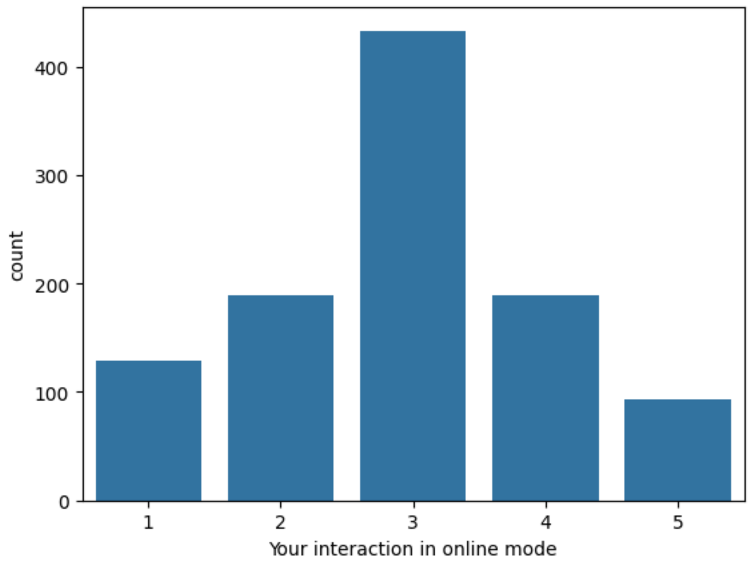

```
#Analise exploratória da variável 5
data['Sleep time (Hours)'].value_counts()
graf = sns.countplot(x ='Sleep time (Hours)', data=data)
```



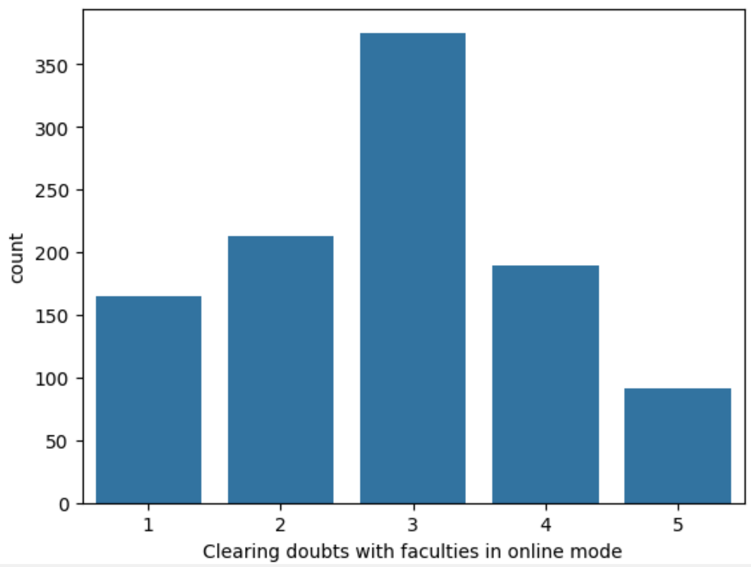

```
#Analise exploratória da variável 6
data['Time spent on social media (Hours)'].value_counts()
graf = sns.countplot(x ='Time spent on social media (Hours)', data=data)
```



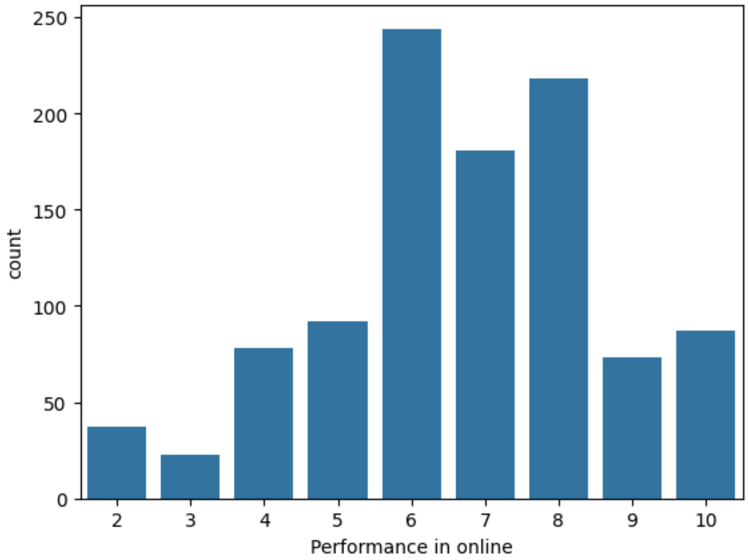

```
#Analise exploratória da variável 7
data['Your interaction in online mode'].value_counts()
graf = sns.countplot(x ='Your interaction in online mode', data=data)
```




```
#Analise exploratória da variável 8
data['Clearing doubts with faculties in online mode'].value_counts()
graf = sns.countplot(x ='Clearing doubts with faculties in online mode', data=data)
```



```
#Analise exploratória da variável 9
data['Performance in online'].value_counts()
graf = sns.countplot(x ='Performance in online', data=data)
```

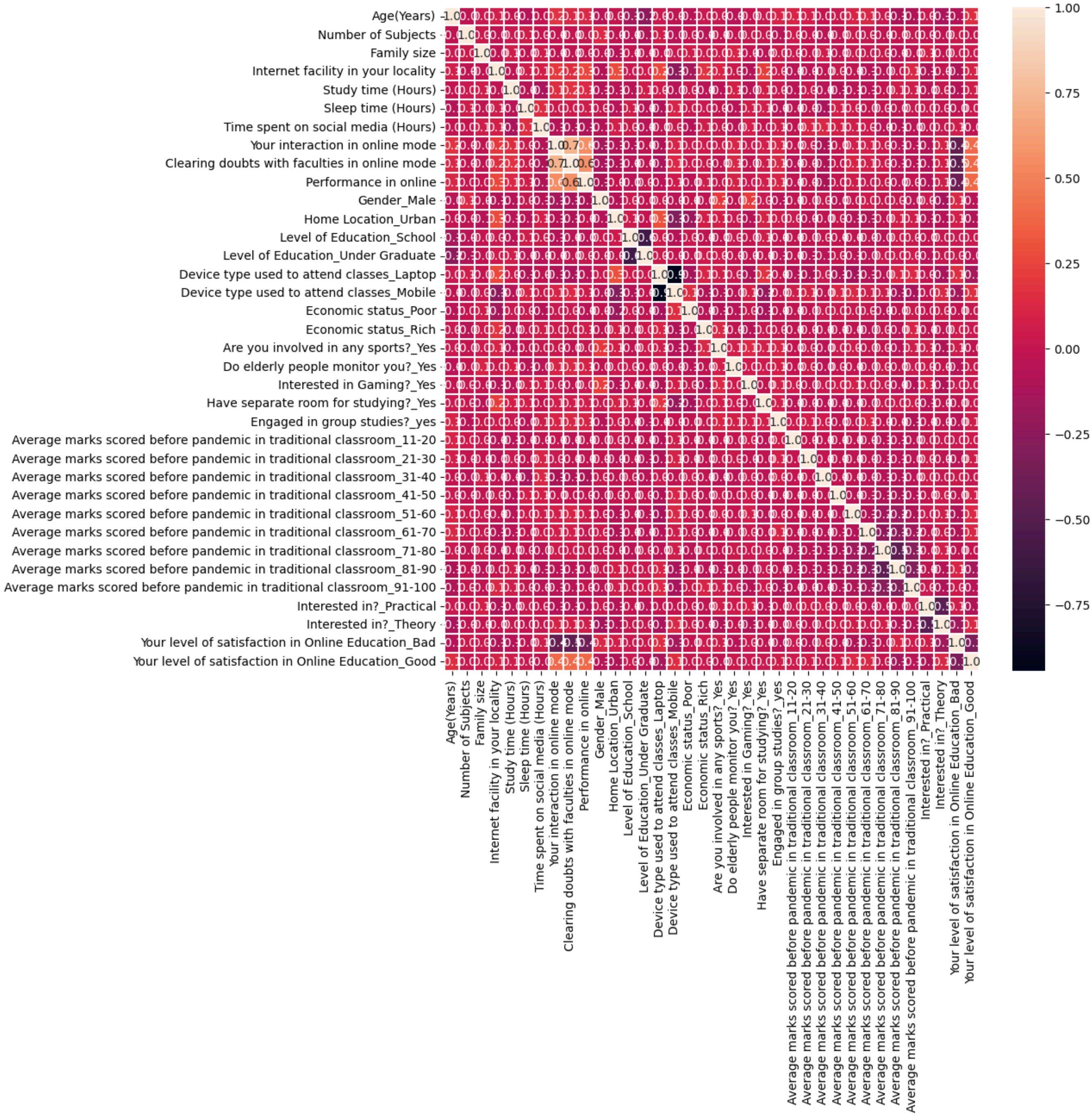



```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Converte variáveis categóricas em variáveis numéricas
data_numeric = pd.get_dummies(data, drop_first=True)

# Mapa de correlação
f, ax = plt.subplots(figsize=(10,10))
sns.heatmap(data_numeric.corr(), annot=True, linewidths=0.05, fmt='.1f', ax=ax)
plt.show()
```



PRÉ PROCESSAMENTO DOS DADOS

*Dados de exemplo de um Dataset

*Dados não minerados

```
#Take the fields of interest and plug them into variable X
x = data[['Age(Years)','Number of Subjects','Family size','Internet facility in your locality','Study time (Hours)','Sleep time (Hours)',
#Make sure to provide the corresponding truth value
y=data['Performance in online'].values.tolist()

#Split the data into test and training (30% for test)
```

```
#Split the data into test and training (30% for test)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

PREDICTION IN MACHINE LEARNING (ML)

```
#Comparando os algoritmos de classificação

#clf = SVC()
#clf = LogisticRegression()
#clf = DecisionTreeClassifier()
#clf = KNeighborsClassifier()
#clf = MLPClassifier()
#clf = RandomForestClassifier()
#clf = GradientBoostingClassifier()
#clf = XGBClassifier()
#clf = LGBMClassifier()


#clf=classificador
```

```
#Criando o classificador com o algoritmo a ser avaliado
clf = LGBMClassifier()
```

```
#Training the classifier using the train data
clf = clf.fit(x_train, y_train)
```

```
[LightGBM] [Warning] Found whitespace in feature_names, replace with underlines
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001155 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 92
[LightGBM] [Info] Number of data points in the train set: 723, number of used features: 9
[LightGBM] [Info] Start training from score -3.287572
[LightGBM] [Info] Start training from score -4.018460
[LightGBM] [Info] Start training from score -2.594425
[LightGBM] [Info] Start training from score -2.472535
[LightGBM] [Info] Start training from score -1.465415
[LightGBM] [Info] Start training from score -1.685569
[LightGBM] [Info] Start training from score -1.514505
[LightGBM] [Info] Start training from score -2.733262
[LightGBM] [Info] Start training from score -2.472535
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
#Avaliation of Machine learning

#validate the classifier

accuracy = clf.score(x_test, y_test)
print('accuracy: ' + str(accuracy))

#Make a confusion matrix

prediction = clf.predict(x_test)

cm = confusion_matrix(prediction,y_test)
cr = classification_report(prediction, y_test)

print(cm)
print(cr)
```

```
accuracy: 0.26129032258064516
[[ 1  1  2  2  3  0  4  0  1]
 [ 0  2  0  0  0  1  1  0  0]
 [ 1  2  3  2  5  3  1  0  1]
 [ 0  0  6  2  4  3  1  2  0]
 [ 4  3  9 13 29  8 10  4  4]
 [ 0  1  2  6 18 14 19  4  4]
 [ 4  1  2  6 17 13 20 11  8]
 [ 0  0  0  0  0  3  1  3  1]
 [ 0  0  0  0  1  2  2  2  7]]
              precision    recall  f1-score   support

           2       0.10      0.07      0.08        14
           3       0.20      0.50      0.29         4
           4       0.12      0.17      0.14        18
           5       0.06      0.11      0.08        18
           6       0.38      0.35      0.36        84
           7       0.30      0.21      0.24        68
           8       0.34      0.24      0.28        82
           9       0.12      0.38      0.18         8
          10       0.27      0.50      0.35        14

    accuracy                           0.26       310
   macro avg       0.21      0.28      0.22       310
weighted avg       0.29      0.26      0.27       310
```

```
from sklearn.metrics import f1_score

# Calculando o F1-score ponderado
y_true = y_test
y_pred = clf.predict(x_test)
f1_weighted = f1_score(y_true, y_pred, average='weighted')

print("F1-score ponderado: ", f1_weighted)
```

```
F1-score ponderado:  0.25567126614902475
```

Resultados

#clf = SVC()

Acuracia
29,03% = low

F1-Score Ponderado 0.41= 41% = low

#clf = LogisticRegression()

Acuracia
28,38% = low

F1-Score Ponderado 0.34= 34% = low

#clf = DecisionTreeClassifier()

Acuracia
18,70% = low

F1-Score Ponderado 0.20= 20% = low

#clf = KNeighborsClassifier()

Acuracia
27,74% = low

F1-Score Ponderado 0.259= 26% = low

#clf = MLPClassifier()

Acuracia
28,70% = low

F1-Score Ponderado 0.258= 26% = low

#clf = RandomForestClassifier()