RAPPORT MACHINE LEARNING

Prédiction des performances des étudiants

Réalisé par:

SABAR Ouafae

KASMI Zoubeir

EL-BOURISSI Hamza

TALAOUI Ghizlane

Encadré par:

Toumi BOUCHENTOUF

Zakaria HAJA

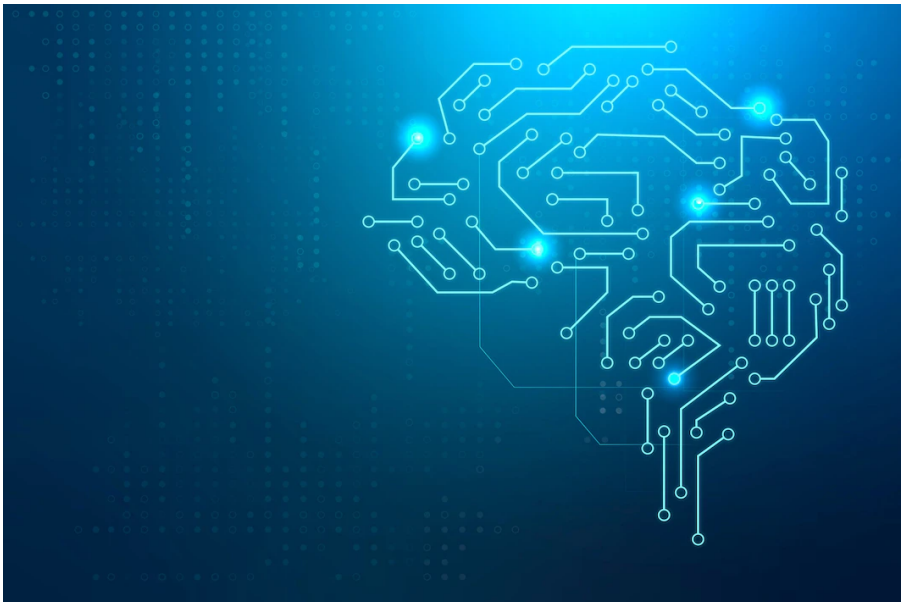
PLAN :

1. Introduction générale
2. Présentation du projet
3. Source des données
4. Preprocessing
5. Machine Learning
6. Déploiement

Introduction :

Le Machine Learning est une branche de **l'intelligence artificielle** qui a pour but **de donner la possibilité aux ordinateurs d'apprendre.** Un ordinateur n'est pas intelligent, il ne fait qu'exécuter des tâches. On lui décrit sous forme de **programmes quoi faire et comment le faire**. C'est ce qu'on appelle la programmation.

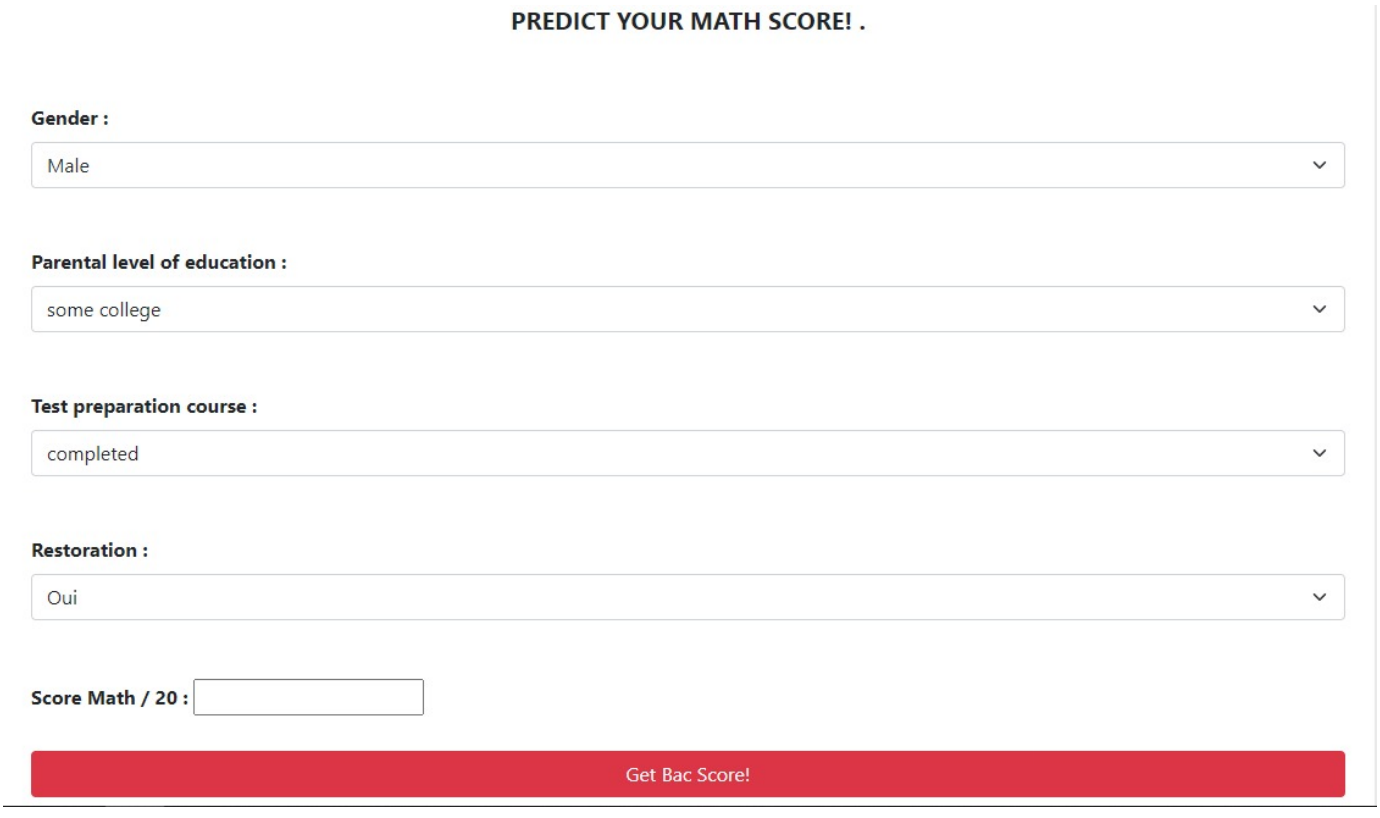
Le machine Learning traite des sujets complexes où la programmation traditionnelle **trouve ses limites**. Construire un programme qui conduit une voiture serait très complexe voire impossible. Cela étant dû aux nombres infinis des cas possibles à traiter… ML traite cette problématique différemment. Au lieu de décrire quoi faire, le programme **apprendra par lui même** comment conduire **en "observant" des expérimentations.**



Présentation du projet :

L’objectif de ce projet est de concevoir et d’implémenter un système d’estimation des notes des étudiants en baccalauréat fondé sur l’apprentissage automatique.

La résolution de cette tâche est fondée sur une chaîne qui a pour entrée le genre de l’etudiant, le niveau d’éducation des parents, la préparation et la resatauration et la moyenne obtenue en devoir des mathématiqes , et a pour sortie la note obtenue en baccalauréat.



Sources des données :

La première étape d'un projet consiste à trouver les datasets et les comprendre.

Les données utilisés dans notre projet sont importés du site Kaggle.

Kaggle est une plateforme web qui accueille la plus grande communauté de Data Science au monde, avec plus de 536 000 membres actifs dans 194 pays et reçoit près de 150 000 soumissions par mois, et qui lui fournit des outils et des ressources puissants pour aider à atteindre tous les progrès de science des données.

Les données sont représentés sous forme de tableau contenant 8 colonnes et sont réunis en fonction des besoins et objectifs du projet.

Preprocessing :

Le preprocessing est l'étape qui consiste à préparer les données avant de les envoyer dans un modèle, afin qu'il puisse travailler convenbalement.

Dans notre cas, il fallait supprimer quelques colonnes :

1. Lire le fichier csv importé de kaggle:

```
Data = pd.read_csv('P.csv')
Data.head()
```

	gender	parents visit	parental level of education	restoration	test preparation course	score math	reading score	score bac
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

2. Eliminer les lignes contenant des valeurs nulles.

```
Data = Data.dropna()
```

3. Supprimer les colonnes reading score et parents visit

```
Data.drop(['reading score','parents visit'],  axis=1, inplace=True)
Data
```

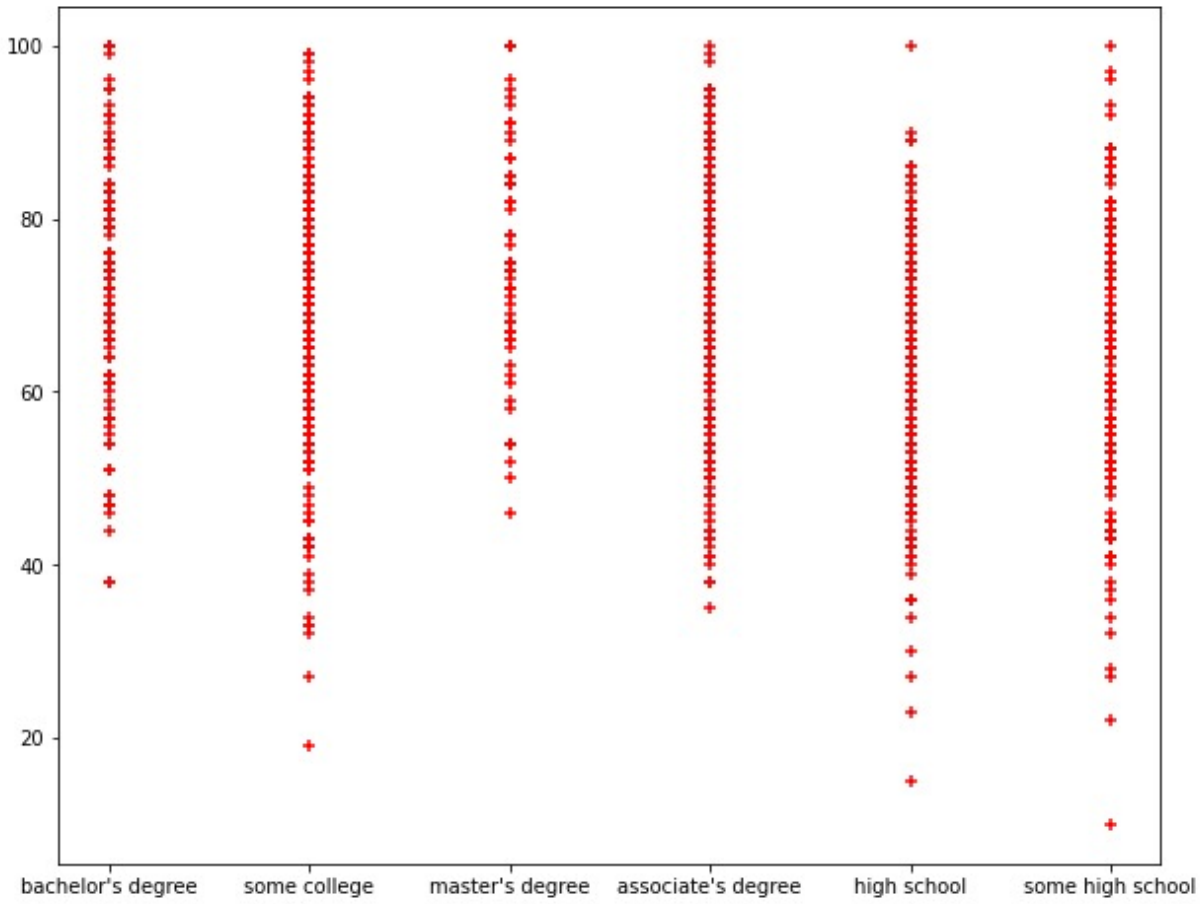
	gender	parental level of education	restoration	test preparation course	score math	score bac
0	female	bachelor's degree	standard	none	72	74
1	female	some college	standard	completed	69	88
2	female	master's degree	standard	none	90	93
3	male	associate's degree	free/reduced	none	47	44
4	male	some college	standard	none	76	75
...	...	...	...	...	...	...
995	female	master's degree	standard	completed	88	95
996	male	high school	free/reduced	none	62	55
997	female	high school	free/reduced	completed	59	65
998	female	some college	standard	completed	68	77
999	female	some college	free/reduced	none	77	86

4. Renommer les labels des colonnes pour qu'elles soient comprehesibles par le programme

```
Index(['gender', 'parental_level', 'restoration', 'test_preparation',
       'score_math', 'score_bac'],
      dtype='object')
```

5. Tracer les nuages des points sur le graphe

```
plt.scatter(Data.gender,Data.score_bac,marker='+',color='red')
plt.ylabel('Score Bac')
```




```
plt.scatter(Data.test_preparation,Data.score_bac,marker='+',color='red')
plt.ylabel('Score Bac')
```

```
Text(0, 0.5, 'Score Bac')
```

```
plt.scatter(Data.restoration,Data.score_bac,marker='+',color='red')
plt.ylabel('Score Bac')
```

```
Text(0, 0.5, 'Score Bac')
```

7. Affecter les colonnes gender, parental_level, restoration et test_preparation aux données d'entrée et le reste aux données de sortie.

```
X = Data.iloc[:,:-1]
Y = Data.iloc[:,-1]
```

8. Convertir les données écrites en valeurs numériques.

```
X['restoration'].replace( 'standard', 0 ,inplace=True)
X['restoration'].replace( 'free/reduced', 1 ,inplace=True)
```

```
X
```

	gender	parental_level	restoration	test_preparation	score_math
0	0	1	0	0	72
1	0	2	0	1	69
2	0	3	0	0	90
3	1	4	1	0	47
4	1	2	0	0	76
...	...	...	...	...	...
995	0	3	0	1	88
996	1	5	1	0	62
997	0	5	1	1	59
998	0	2	0	1	68
999	0	2	1	0	77

9. Diviser les données en dataset et trainset (20%).

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)
```

Machine Learning :

Puisque nous travaillons avec un problème de regression nous utiliserons les modèles **Regression linéaire** et **Regression Lasso**.

1. Regression linéaire

La régression linéaire est un modèle statistique spécialisé dans **la mise en œuvre des fonctions prédictives** avec un minimum d'erreurs. Cet algorithme exploite des valeurs numériques pour dégager une tendance ou une évolution prévisible dans le temps.

1. On a importé la classe LinearRegressor, l'instancier et appeler la méthode fit() avec nos données de formation.

```
regressor = LinearRegression()

regressor.fit(X_train,Y_train)
```

```
LinearRegression()
```

2. On compare les prédictions calculés par le programme avec les données qu'on a déja

```
Y_test.to_numpy()
```

```
array([ 62,  58,  69,  83,  75,  79,  54,  75,  65,  64,  48,  83,  69,
        75,  52,  59,  78,  73,  70,  74,  51,  67,  54,  62,  72,  77,
        34,  64,  71,  48,  75,  67,  54,  87,  78,  65,  87,  87,  58,
        52,  78,  69,  72,  76,  72,  64,  55,  74,  69,  98,  58,  73,
        68,  77,  72,  55,  68,  75,  68,  51,  96, 100,  57,  54,  74,
        64,  88,  69,  73,  80,  75,  65,  75,  42,  75,  50,  62,  65,
        99,  64,  73,  60,  86,  53,  70,  59,  47,  60,  72,  77,  66,
        69,  52,  91,  66,  51,  66,  72,  61,  80,  47,  50,  50,  82,
        90,  79,  70,  44,  32,  59,  69,  58,  49,  51,  63,  74,  66,
        53,  85,  58,  81,  74,  94,  51,  68,  70,  66,  80,  76,  53,
        27,  76,  74,  48,  56,  82,  91,  82,  73,  84,  97,  73,  60,
        78,  83,  66,  85,  94,  60,  33,  42,  70,  54,  61,  57,  95,
        62,  23,  51,  36,  78,  81,  38,  55,  60,  75,  69,  85,  86,
        95,  38,  59,  34,  51,  70,  83,  22,  52,  74,  55,  73,  64,
        79,  71,  74,  70,  80,  58,  70,  68,  47, 100,  88,  84,  72,
        72,  57,  67,  73,  64], dtype=int64)
```

```
Y_pred=regressor.predict(X_test)
```

```
Y_pred
```

```
array([ 63.8781001 ,  58.02350608,  62.47567885,  84.24797439,
        76.46281027,  85.22351978,  52.37788026,  74.78157674,
        69.84929544,  54.74938921,  47.84945679,  79.4080777 ,
        59.10799642,  72.74006486,  59.89515064,  64.68377815,
        76.45512057,  67.58669813,  59.25605999,  67.97319003,
        49.79412321,  67.31880438,  56.25634562,  55.59427028,
        77.90881099,  75.2770136 ,  38.61183427,  62.20455627,
        75.37831993,  46.35664776,  76.72624315,  62.69999312,
        58.44269224,  87.09516433,  80.38808395,  55.49296395,
        84.51140727,  72.86312387,  64.46392474,  58.07026332,
        77.70302056,  59.25605999,  65.07150207,  81.78281549,
        66.45093861,  62.42887054,  55.81535571,  68.7648051 ,
        57.90367591,  87.75401083,  65.05620707,  73.55666145,
        81.24826002,  74.14571494,  78.98561163,  62.47567885,
        68.39560502,  81.46934545,  63.40887678,  50.38640553,
        92.10492835, 107.31697506,  63.27812807,  60.38735865,
        70.43388807,  61.51096761,  80.43807003,  72.87081357,
        67.90462903,  82.64299156,  76.62621992,  62.96019717,
        70.55371824,  53.87391814,  73.50344653,  61.83981705,
        62.64226626,  61.61550278, 101.29574255,  63.72680768,
        60.55399714,  57.31462242,  90.39226591,  56.62103376,
```

3. On mesure le presicion du modele

```
regressor.score(X_test,Y_test)
```

```
0.8708138357655084
```

Ici, nous avons obtenu un score de précision de **87 %** sur l'ensemble de données de test.

Exemple ordinaire pour le test:

```
regressor.predict([[0,1,1,1,60]])
```

```
array([75.02981556])
```

2. Regression Lasso

Cette technique est un type de régression linéaire et permet de réduire les limites du modèle. Les valeurs des données se réduisent au centre ou à la moyenne pour éviter de surcharger les données. En utilisant le contexte de la régression de crête, nous comprendrons cette technique en détail ci-dessous en utilisant des mots simples.

1. On a importé la classe Lasso, l'instancier et appeler la méthode fit() avec nos données de formation.

```
model = Lasso()
model.fit(X_train,Y_train)
```

```
Lasso()
```

2. On compare les prédictions calculés par le programme avec les données qu'on a déja

```
: Y_pred=model.predict(X_test)
```

```
: Y_pred
```

```
array([62.79454832, 75.58280753, 71.18173907, 57.90438642, 52.88680919,
       52.30760464, 53.3142664 , 84.67224305, 63.32485993, 71.79778669,
       62.76975515, 69.87503542, 37.02130356, 73.75784216, 69.23373348,
       74.56340248, 58.58253142, 68.50232043, 69.15567226, 61.87799767,
       61.64005317, 84.62288896, 78.12206756, 51.61740975, 80.91247653,
       82.05883576, 62.49450644, 54.72382445, 79.22766977, 56.89772465,
       72.72638722, 67.43402243, 98.94374644, 98.90690338, 73.94689372,
       80.49706921, 59.24793313, 33.0103881 , 70.19917708, 67.5737199 ,
       95.57413291, 77.933016  , 73.67164501, 84.39699434, 48.05874467,
       56.67183004, 71.38284051, 61.310843  , 61.65210305, 56.39658133,
       91.17306445, 96.88083656, 75.85805624, 64.64361347, 65.93826723,
       46.76409091, 79.1535225 , 65.85207007, 61.27399994, 82.37092753,
       48.16159907, 71.0047374 , 55.37717627, 74.5513526 , 67.86101849,
       81.94393147, 61.48784467, 74.56340248, 56.01847821, 81.8697842 ,
```

```
Y_test.to_numpy()
```

```
array([ 65,  84,  68,  53,  58,  40,  51,  82,  62,  77,  73,  73,  27,
        81,  62,  73,  56,  68,  59,  66,  58,  89,  78,  44,  75,  87,
        65,  54,  70,  53,  78,  67, 100,  96,  81,  79,  60,  32,  79,
        74, 100,  84,  81,  89,  41,  46,  70,  62,  56,  57,  94, 100,
        61,  68,  51,  43,  82,  57,  62,  98,  46,  81,  55,  69,  74,
        79,  63,  77,  52,  77,  92,  79,  71,  74,  97,  60,  45,  78,
        72,  72,  76,  43,  65,  79,  45,  65,  43,  75,  65,  63,  50,
        19,  49,  72,  67,  57,  87,  53,  69,  67,  84,  56,  78,  68,
        62,  54,  73,  70,  69,  63,  75,  63,  68,  55,  92,  60,  38,
        48,  82,  88,  85,  86,  76,  80,  79,  91,  50,  71,  70,  60,
        73,  78,  87,  66,  43,  54,  69,  81,  72,  59,  90,  86,  47,
       100,  61,  47,  78,  71,  41,  83,  54,  66,  67,  66,  70,  55,
        76,  51,  75,  73,  54,  79,  74,  64,  85,  68,  68,  65,  72,
        66,  46,  68,  75,  56,  52,  41,  61,  51,  92,  78,  72,  76,
        63,  69,  84,  67,  80,  57,  66,  62,  74,  61,  72,  68,  50,
        88,  76,  54,  49,  67], dtype=int64)
```

3. On mesure le presicion du modele

```
model.score(X_test,Y_test)
```

```
0.8144221146536349
```

Dans ce cas, nous avons obtenu un score de précision de **81 %** sur l'ensemble de données de test.

Exemple ordinaire pour le test:

```
model.predict([[0,1,1,1,60]])
```

```
array([68.77817859])
```

On valide donc le modèle de regression linéaire.

Déploiement :

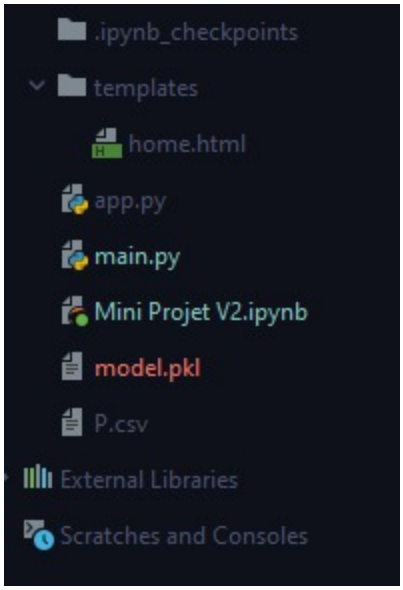
Après le traitement des données et le choix du modèle final (Regression linéaire) qui a donné la meilleure précision, il faut déployer notre modèle machine learning en production. Pour cela on s'est servi du PyCharm et Flask.

Pycharm est un éditeur de code intelligent qui fournit un support de première classe pour Python, JavaScript, CoffeeScript, TypeScript, CSS, le langage de modèle populaire et plus encore. Tirez parti de la complétion de code, de la détection des erreurs et des corrections de code à la volée!

Flask est un petit framework web Python léger, qui fournit des outils et des fonctionnalités utiles qui facilitent la création d'applications web en Python. Il offre aux développeurs une certaine flexibilité et constitue un cadre plus accessible pour les nouveaux développeurs puisque vous pouvez construire rapidement une application web en utilisant un seul fichier Python. Flask est également extensible et ne force pas une structure de répertoire particulière ou ne nécessite pas de code standard compliqué avant de commencer.

Flask

Dans notre cas :



main.py : code source de l’application.

model.pkl : le fichier générer.

app.py : code qui lie l’interface d'utilisateur avec notre model.

home.html : interface d'utilisateur.