ÉTUDE DE PROJET MACHINE LEARNING

BOUGUERRA Mhamed

2MP-SD

I) Analyse de besoins

1) Définition du programme

Note projet et définir comme un système d'apprentissage (Analyse et prédiction) qui permet de découvrir le nombre des personnes qui se font vacciner avec une data bien déterminer aussi prédire le nombre de personnes qui se font vacciner à cette date ainsi qu'à un autre moment.

2) Exigences

- Le logiciel doit être capable de s'entraîner avec des données pour atteindre le taux de prédiction le plus élevé.
- Le logiciel doit être recyclé à l'aide des données.
- Le logiciel doit avoir une interface graphique facile à comprendre pour l'utilisateur final.
- La précision doit atteindre au moins 70%+.

3) Solution proposée

Le but de l'application (Tunisiens Vac) est découvrir le nombre de personnes qui se font vacciner à la date que nous avons choisie (lorsque nous sommes à la date et le nombre de personnes inscrites, cela nous prédire le nombre de personnes qui se font vacciner à cette date, même la date est dans le futur).

Pour ce faire, l'application doit permettre les fonctionnalités suivantes :

- Charger les données.
- Démarrez l'analyse.
- Train le model.

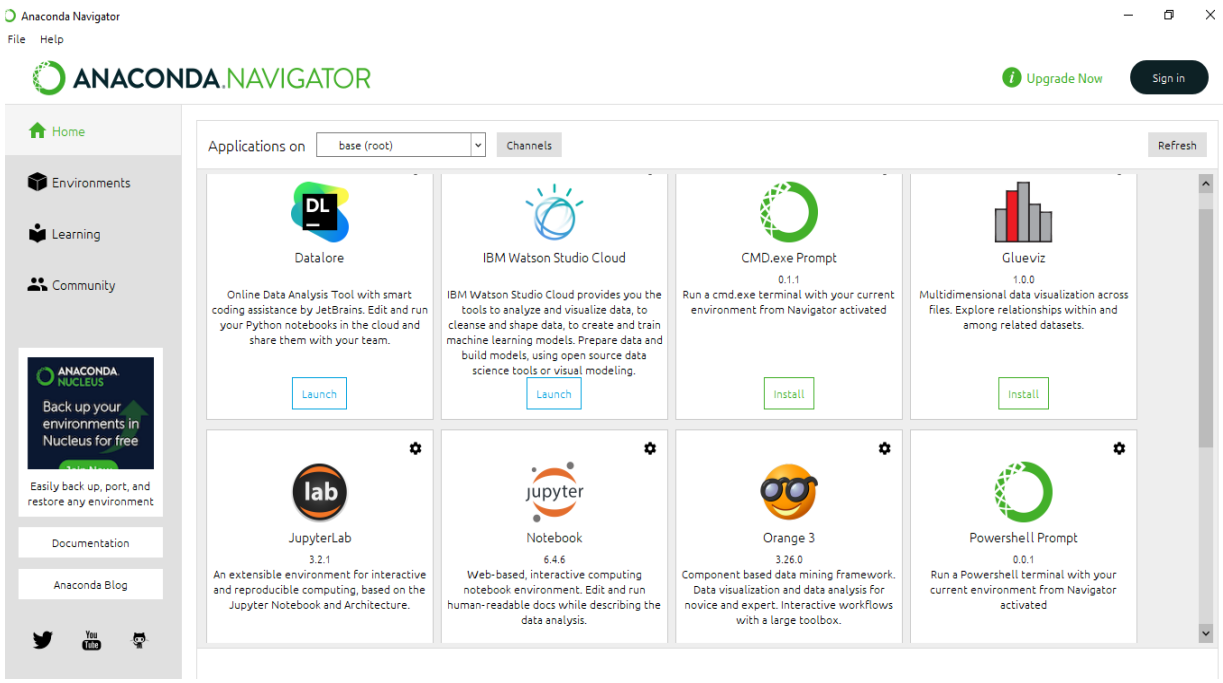
II) Méthode de conception

1) Environnement logiciel

Dans cette phase, nous avons montré les différents outils utilisé tout au long de ce projet pour la mise en place de notre application.

1.2) Environnement de développements

Dans ce cadre-là on va utiliser « Anaconda » comme une plateforme de distribution des langages de programmation Python et R pour le calcul scientifique (science des données, applications d'apprentissage automatique, traitement de données à grande échelle, analyse prédictive, etc.), qui vise à simplifier la gestion et le déploiement des packages.



2) Base de Données

Notre dataset est fondamentalement un fichier du type csv. La source de ce dernier est la plateforme sanitaire EVAX.

3) Environnement matériel

Nous mentionnons les caractéristiques de notre ordinateur sur lesquelles nous avons développé l'application. Cette dernière a été développée sur une machine avec les caractéristiques suivantes :

- Marque : ACER Aspire F5-573G
- Processeur : Intel(R) Coré (TM) i7-7500U CPU @ 2.70GHz 2.90 GHz
- Mémoire RAM : 12.00 Gb.
- Système d'exploitation : Windows 10 Professionnel.

III) Simulation

1) Description de l'application

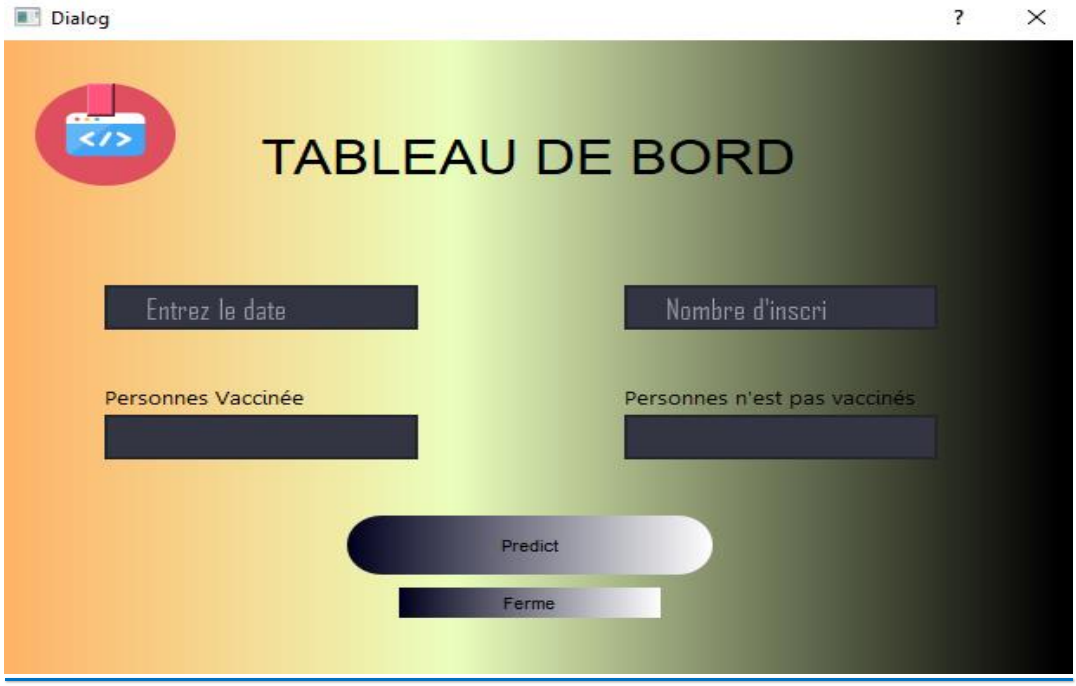
1.1) Présentation des interfaces de l'application

a) Interface de Bienvenue

Dans notre première interface nous avons une interface de bienvenue contient le nom de projet avec le name de réalisateur et le choix pour entrer ou bien quitté l'application

b) Interface du Tableau de bord

Dans cette interface nous permettent de faire des prédictions sur notre dataset. On va utiliser l'algorithme de modèle d'apprentissage automatique supervisé : Linear Régression.



c) Interface du lien Source code

Avec cette button notre interface en va nous transfert dans le lien de code source de notre application.

Captures d’écran du code source

- Importer des bibliothèques qui on va travailler avec

Tunisian Vaccine

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

- Importer de la base de données et afficher les 5 premières lignes

```
data = pd.read_csv('https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/vaccinations/country_data/Tunisia.csv
data.head(5)
```

	location	date	vaccine	source_url	total_vaccinations	people_vaccinated	people_fully_vaccinated	total_boosters
0	Tunisia	2021-03-12	Sputnik V	https://www.africanews.com/2021/03/13/tunisia-...	0	0	0.0	NaN
1	Tunisia	2021-03-13	Sputnik V	https://www.facebook.com/santetunisie.rns.tn/v...	743	743	0.0	NaN
2	Tunisia	2021-03-14	Sputnik V	https://www.facebook.com/santetunisie.rns.tn/p...	2076	2076	0.0	NaN
3	Tunisia	2021-03-15	Sputnik V	https://www.facebook.com/186480324724413/posts...	2555	2555	0.0	NaN
4	Tunisia	2021-03-19	Sputnik V	https://www.facebook.com/186480324724413/posts...	6861	6861	0.0	NaN

- Supprimer les colonnes que nous n'en avons pas besoin et afficher la résultat

```
data= data.drop(['vaccine','source_url','location','total_boosters'], axis=1)
data.head(5)
```

	date	total_vaccinations	people_vaccinated	people_fully_vaccinated
0	2021-03-12	0	0	0.0
1	2021-03-13	743	743	0.0
2	2021-03-14	2076	2076	0.0
3	2021-03-15	2555	2555	0.0
4	2021-03-19	6861	6861	0.0

- Afficher les données disparus (NULL)

```
data.isnull().sum()
```

```
date                        0
total_vaccinations          0
people_vaccinated           0
people_fully_vaccinated     3
dtype: int64
```

- Replacer tous les cas nuls avec 0 pour éviter les problèmes dans la partie prédiction

```
data['people_fully_vaccinated'] = data['people_fully_vaccinated'].fillna(0)
data[data.isnull().any(axis=1)]
```

date	total_vaccinations	people_vaccinated	people_fully_vaccinated

- Afficher la description de notre dataset

```
data.describe()
```

	total_vaccinations	people_vaccinated	people_fully_vaccinated
count	2.270000e+02	2.270000e+02	2.270000e+02
mean	4.518149e+06	2.844926e+06	1.997091e+06
std	3.938548e+06	2.340088e+06	1.986165e+06
min	0.000000e+00	0.000000e+00	0.000000e+00
25%	8.204935e+05	5.512720e+05	2.487035e+05
50%	2.619884e+06	1.681477e+06	7.814830e+05
75%	8.599588e+06	5.355410e+06	4.140680e+06
max	1.105785e+07	6.312309e+06	5.377874e+06

- Afficher la corr de notre dataset
- Afficher le nombre du colonne et du ligne de notre dataset

```
data.corr()
```

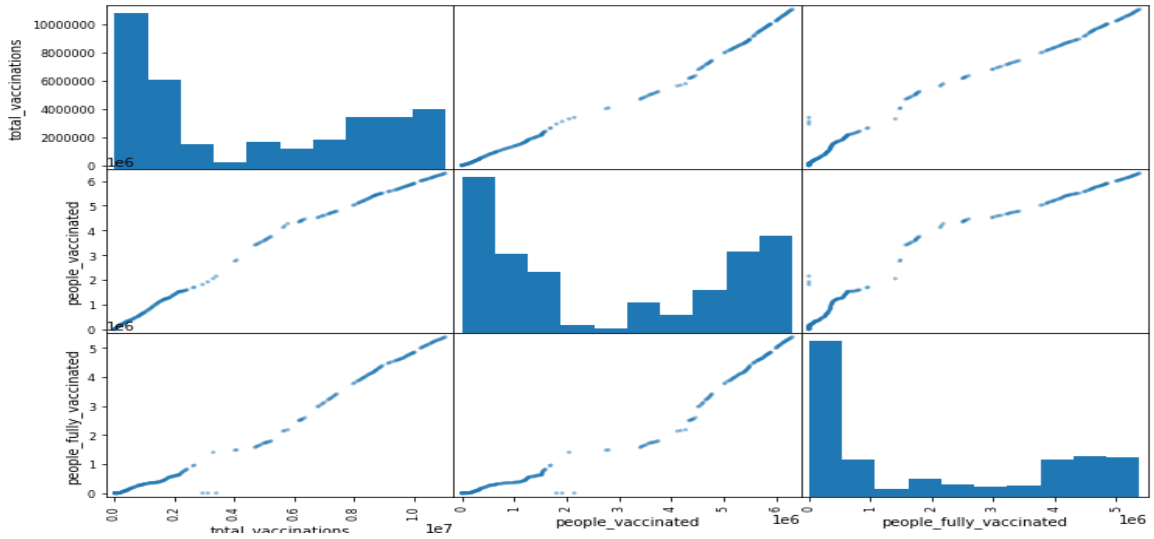
	total_vaccinations	people_vaccinated	people_fully_vaccinated
total_vaccinations	1.000000	0.995685	0.992193
people_vaccinated	0.995685	1.000000	0.980617
people_fully_vaccinated	0.992193	0.980617	1.000000

```
data.shape
```

```
(227, 4)
```

- Importer bibliothèque scatter_matrix et représenter la distribution des données

```
from pandas.plotting import scatter_matrix
ls = ["date","total_vaccinations", "people_vaccinated","people_fully_vaccinated"]
scatter_matrix(data[ls], figsize=(12, 8));
```



- Afficher les informations et les types des colonnes

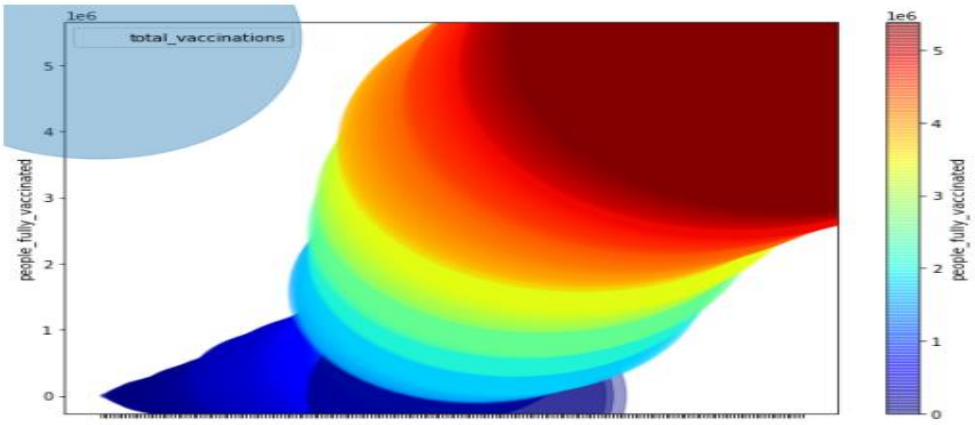
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 4 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   date                     227 non-null    object 
 1   total_vaccinations       227 non-null    int64  
 2   people_vaccinated        227 non-null    int64  
 3   people_fully_vaccinated  227 non-null    float64
dtypes: float64(1), int64(2), object(1)
memory usage: 7.2+ KB
```

- Afficher une courbe du nombre des personnes vaccinés faible jusqu'au devient plus nombreux

```
data.plot(kind="scatter", x="date", y="people_fully_vaccinated", alpha=0.4,
  s=data["total_vaccinations"]/100, label="total_vaccinations", figsize=(10,7),
  c="people_fully_vaccinated", cmap=plt.get_cmap("jet"), colorbar=True,
)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x148e0646b20>
```



- Changer le type de date en numérique pour éviter les problèmes de train de model

Parti Machine Learning

```
data['date'] = pd.to_numeric(data.date.str.replace('-',''))
data.head()
```

	date	total_vaccinations	people_vaccinated	people_fully_vaccinated
0	20210312	0	0	0.0
1	20210313	743	743	0.0
2	20210314	2076	2076	0.0
3	20210315	2555	2555	0.0
4	20210319	6861	6861	0.0

- Diviser notre dataset en deux parties x et y

```
x=data[['date','total_vaccinations']].values
y=data[['people_vaccinated']].values
```

- Importer la bibliothèque sklearn et fit le dataset avec le model LinearRegression

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x,y)
```

```
LinearRegression()
```

- Importer la bibliothèque train_test_split et fit un deuxième model de test avec le model LinearRegression

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
model2 = LinearRegression()
model2.fit(x_train,y_train)
```

```
LinearRegression()
```

- Afficher le score du notre test dataset qui est égale a 98,99%

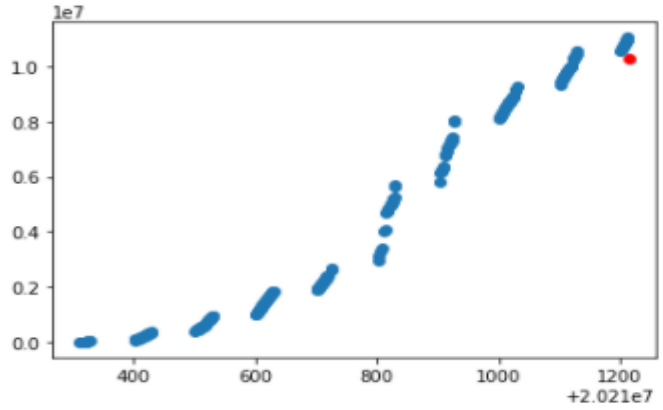
```
score=(model2.score(x_test,y_test))*100
score
```

```
98.99771293375608
```

- Pointer dans le modèle scatter un point que nous voulons savoir leur prédiction

```
plt.scatter(x[:,0],x[:,1])
plt.scatter(20211214,10292254,c='red')
```

```
<matplotlib.collections.PathCollection at 0x148e2ca60a0>
```



- Faire la prédiction d'une random saisir (dans la future)

```
pred=model.predict([['20211214','10292254']])
print('La Résultat est:',int(pred))
```

```
La Résultat est: 6309914
```

- Parti interface qui été créé avec Qt_Designer

Parti Interface

```
# importation des bibliothèques des tools QTWidgets
from PyQt5 import QtWidgets, uic
import sys
import webbrowser
```

```
# ----------------------functions Front-------------------------
def exit():
    Fen.destroy()
    Fen2.destroy()
def transfert():
    Fen2.show()
    Fen.destroy()
# ----------------------functions Back-------------------------
def lien():
     webbrowser.open_new('http://localhost:8888/notebooks/Desktop/WORK2/Projet_ML_MR_MED_Kharrat.ipynb')
def trait():
    d=Fen2.lineEdit.text()
    i=Fen2.lineEdit_2.text()
    rt=model.predict([[d,i]])
    Fen2.lineEdit_3.setText(str(int(rt)))
    rt2=11935764-rt
    Fen2.lineEdit_4.setText(str(int(rt2)))
```

```
App =QtWidgets.QApplication(sys.argv)
#  ---------------Front----------------
Fen=uic.loadUi('Front.ui')
Fen.show()
Fen.pushButton.clicked.connect(transfert)
Fen.pushButton_2.clicked.connect(exit)
# -------------Back--------------
Fen2=uic.loadUi('Back.ui')
Fen2.pushButton.clicked.connect(trait)
Fen2.pushButton_2.clicked.connect(lien)
Fen2.pushButton_3.clicked.connect(exit)
# -----------Exécutér----------
App.exec_()
sys.exit()
```