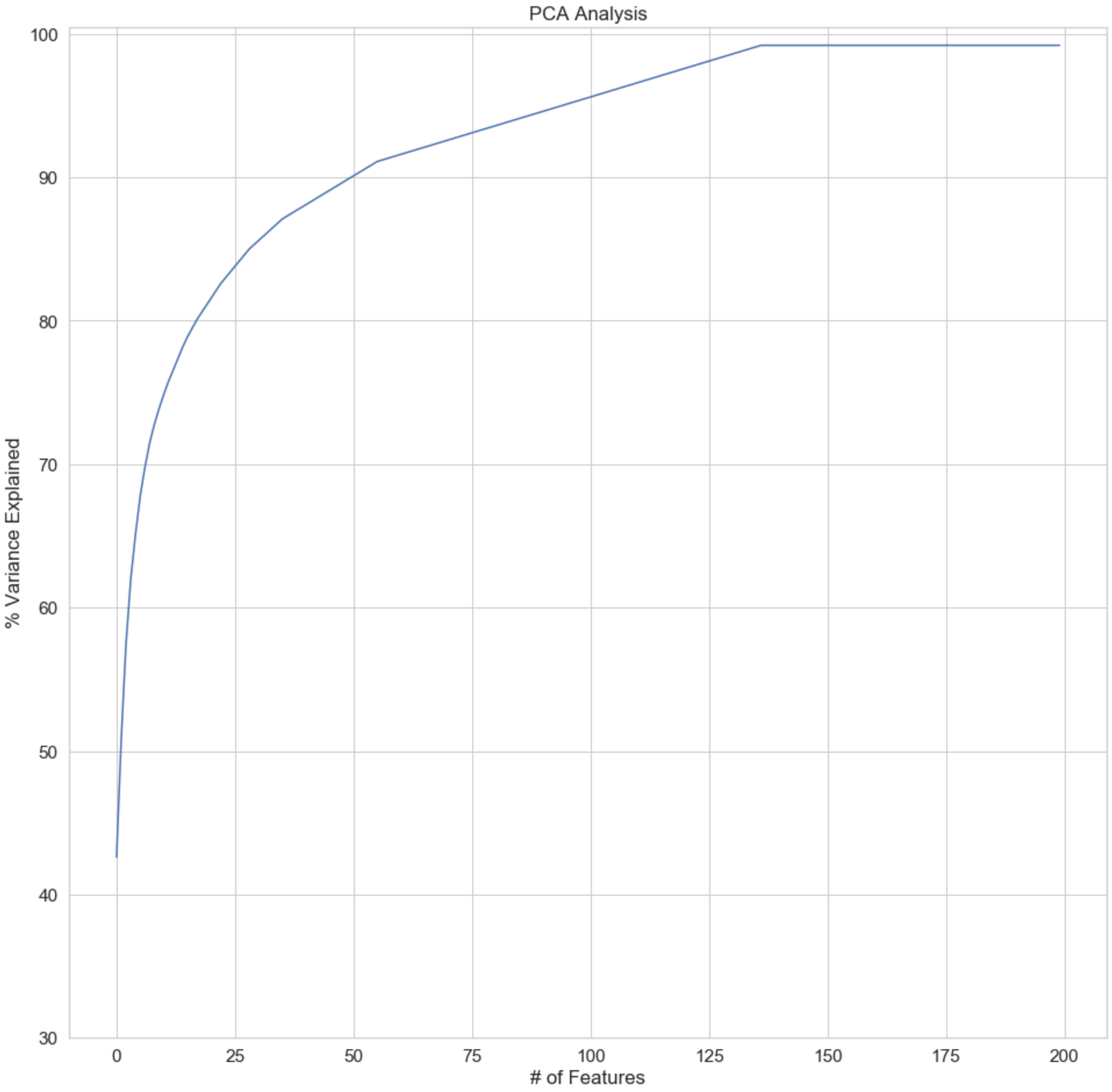
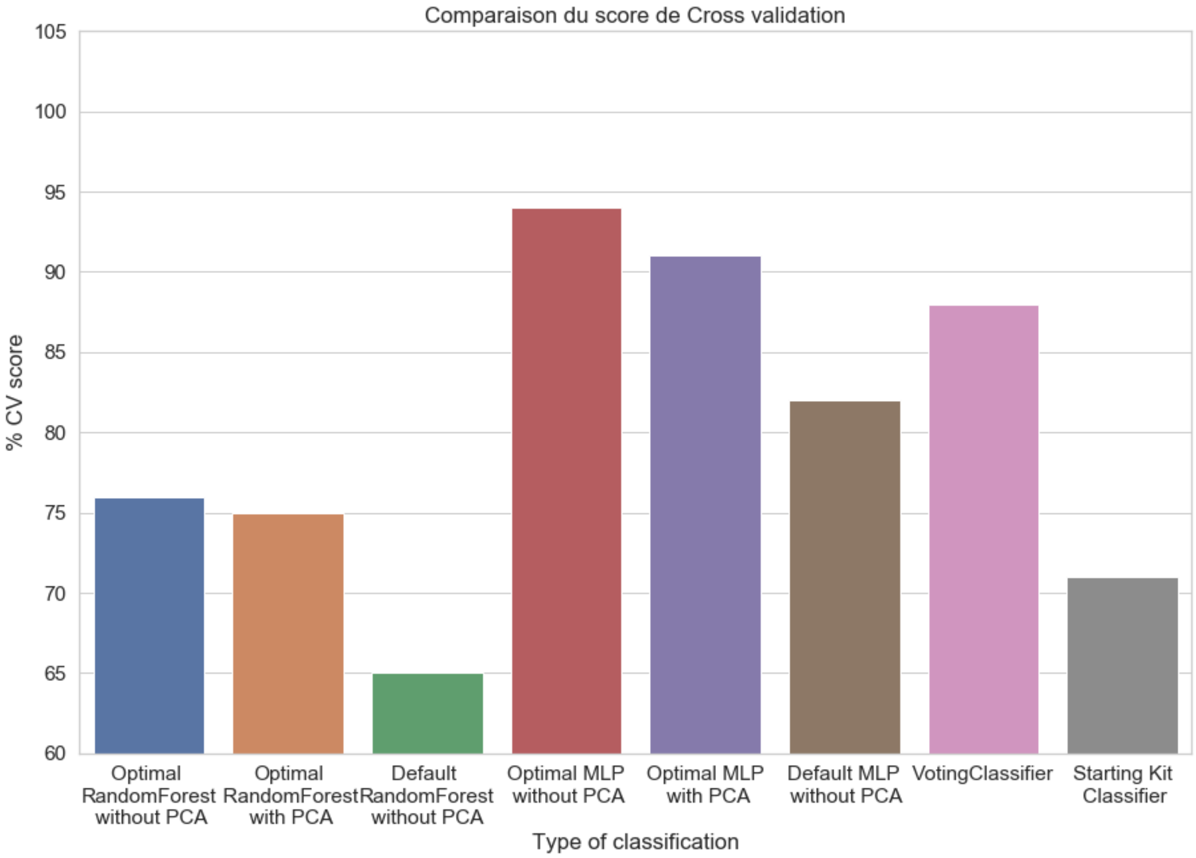
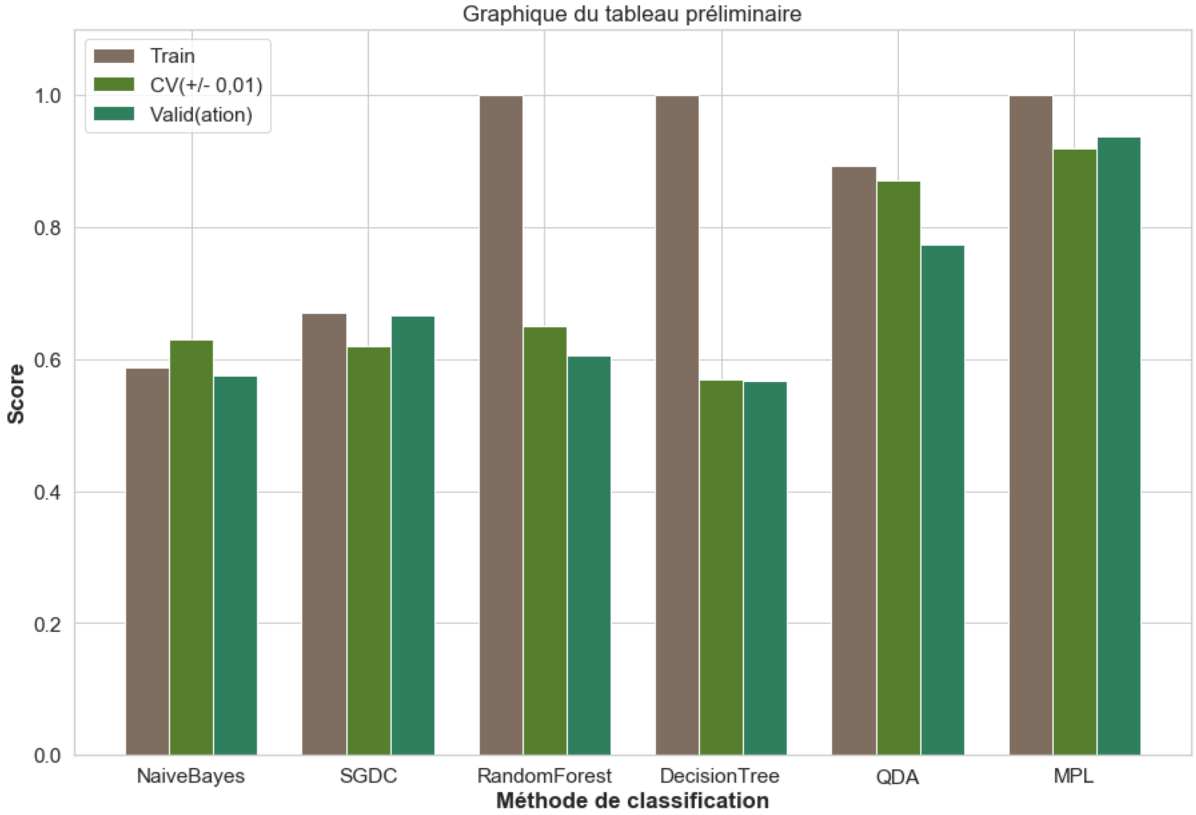
VERSION DU 25/04/19

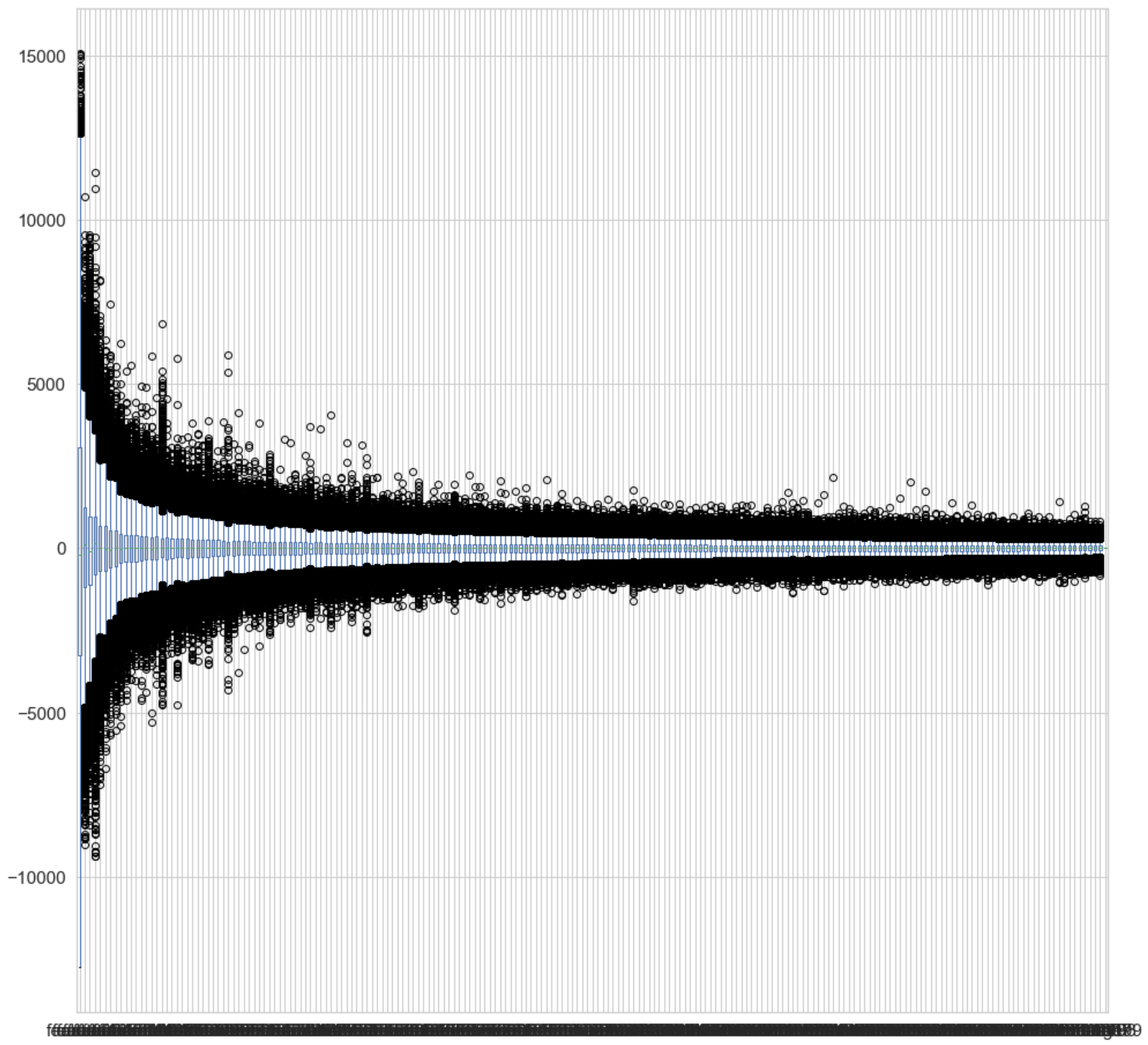
SORTIE CONSOLE :

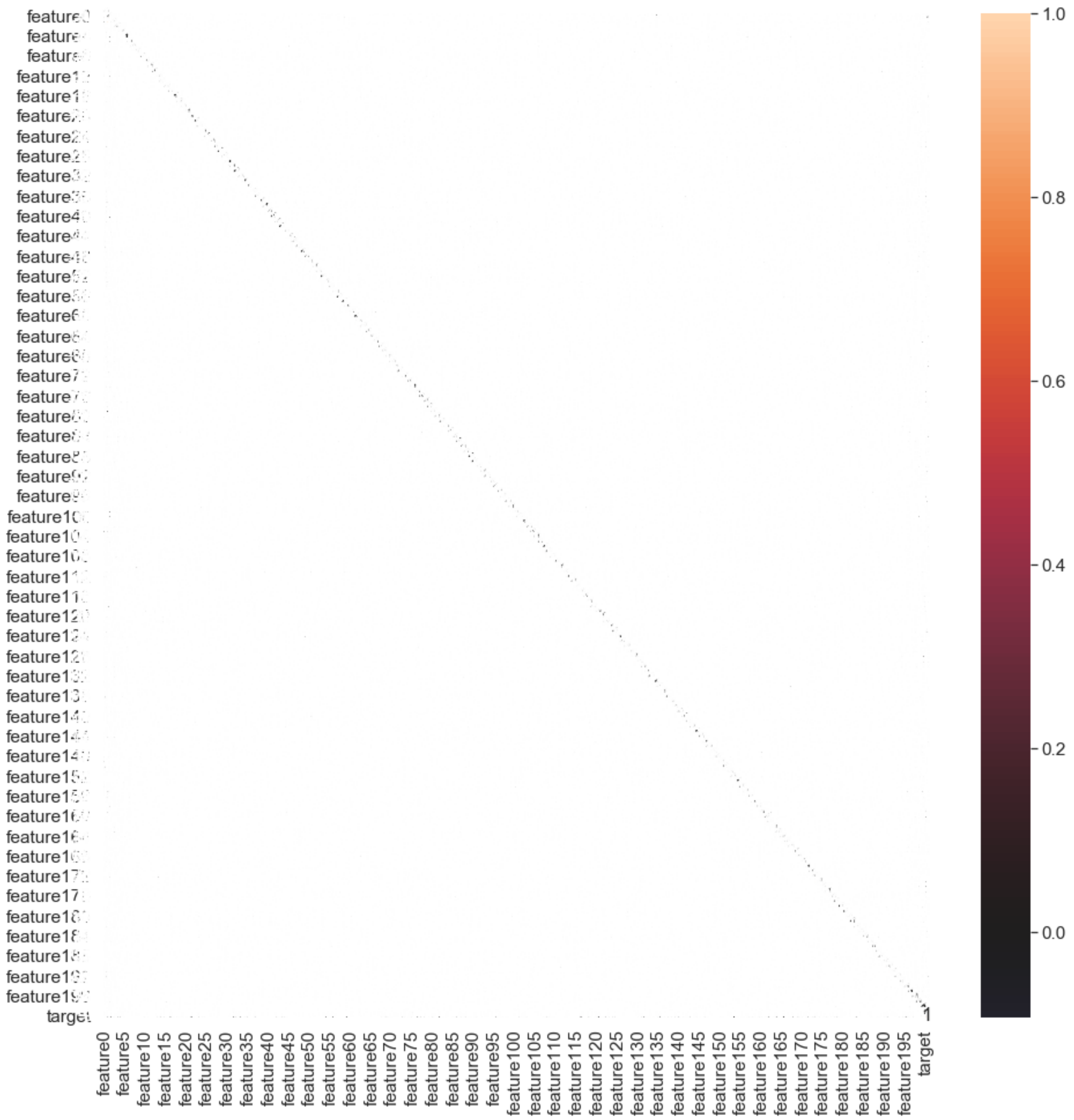
```
runfile('C:/Users/isabe/Pictures/monetFINAL/starting_kit/sample_code
_submission/visualisation.py',
wdir='C:/Users/isabe/Pictures/monetFINAL/starting_kit/sample_code_su
bmission')
Reloaded modules: data_io, data_converter
Reading C:\Users\isabe\Downloads\monet-
master\starting_kit\c1_input_data\perso_train from AutoML format
Number of examples = 65856
Number of features = 200
   Class
0  False
1   True
Number of classes = 2
```



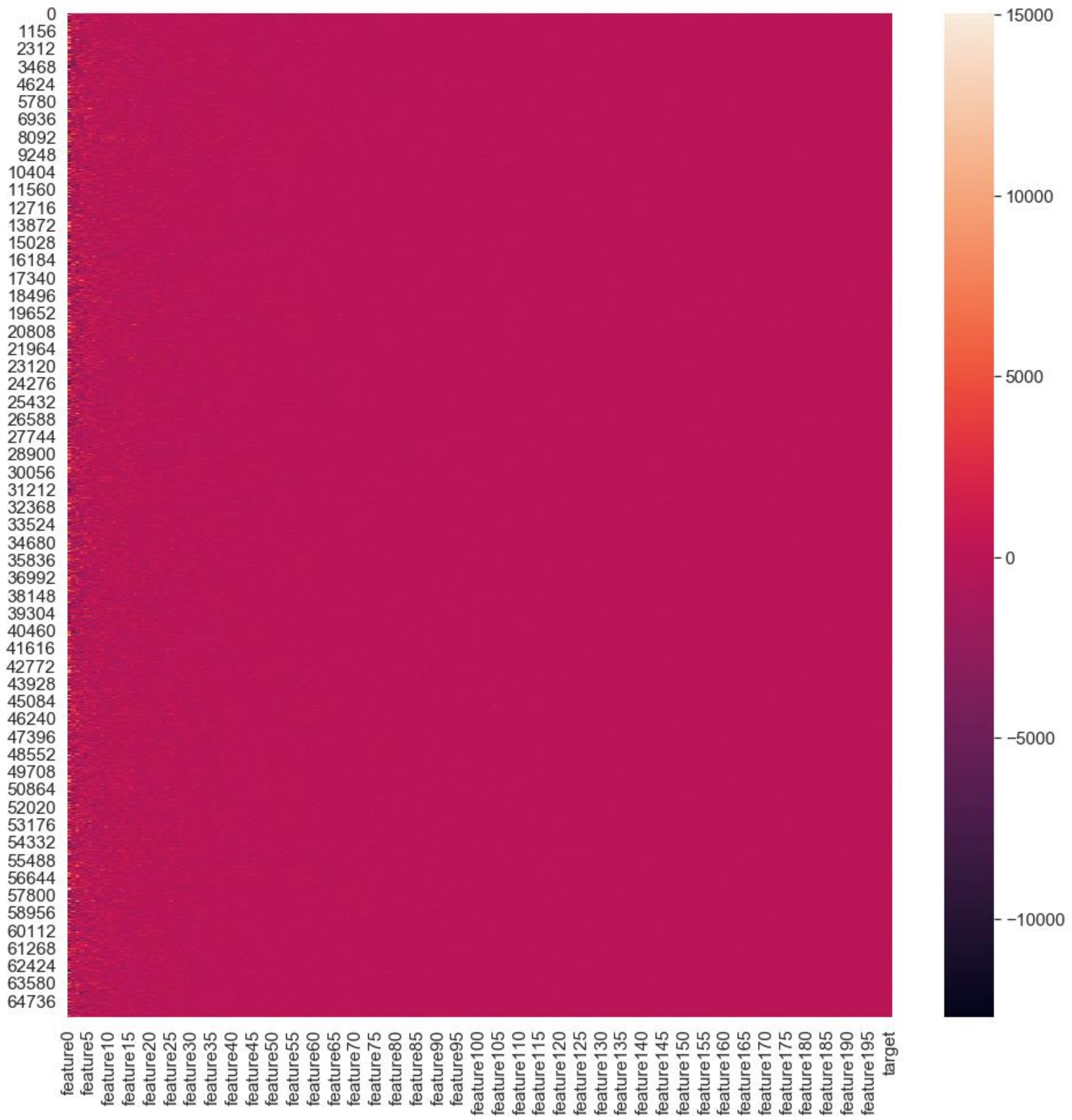

Comparaison du score de Cross validation
% CV score
105
100
95
90
85
80
75
70
65
60
Optimal RandomForest without PCA
Optimal RandomForest with PCA
Default RandomForest without PCA
Optimal MLP without PCA
Optimal MLP with PCA
Default MLP without PCA
VotingClassifier
Starting Kit Classifier
Type of classification


Graphique du tableau préliminaire
Train
CV(+/- 0,01)
Valid(ation)
Score
1.0
0.8
0.6
0.4
0.2
0.0
NaiveBayes
SGDC
RandomForest
DecisionTree
QDA
MPL
Méthode de classification


15000
10000
5000
0
−5000
−10000



feature0
feature5
feature10
feature15
feature20
feature25
feature30
feature35
feature40
feature45
feature50
feature55
feature60
feature65
feature70
feature75
feature80
feature85
feature90
feature95
feature100
feature105
feature110
feature115
feature120
feature125
feature130
feature135
feature140
feature145
feature150
feature155
feature160
feature165
feature170
feature175
feature180
feature185
feature190
feature195
target
1
1.0
0.8
0.6
0.4
0.2
0.0



CODE :

```
CODE
# -*- coding: utf-8 -*-
"""
Created on Fri Feb 22 12:30:02 2019
@author: isabe
"""

#Imports
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
```

```
import seaborn as sns
import data_io


'''Il s'agit d'un fichier permettant la generation d'images'''


def funcPCA (data) :

    #print(np.cumsum(pca.explained_variance_ratio_))
    plt.figure(figsize = (16,16))
    plt.ylabel('% Variance Explained')
    plt.xlabel('# of Features')
    plt.title('PCA Analysis')
    plt.ylim(30,100.5)
    plt.plot(np.cumsum(np.round(pca.explained_variance_ratio_, 
decimals=3)*100))
    plt.show()
    plt.close()


def funcDiagbaton (donnee) :
    plt.figure(figsize = (15,10))




    if( donnee==1):
        plt.ylim(60,105)
        plt.title('Comparaison du score de Cross validation')
        fake = pd.DataFrame({'Type of classification': ['Optimal
\nRandomForest\n without PCA', 'Optimal\n RandomForest\n with PCA', 
'Default \nRandomForest\nwithout PCA', 'Optimal MLP\n without PCA', 
'Optimal MLP\n with PCA', 'Default MLP\n without PCA', 
'VotingClassifier', 'Starting Kit\n Classifier'], '% CV score': [76, 
75,65, 94, 91, 82,88, 71]})
        fig = sns.barplot(x = 'Type of classification', y = '% CV 
score', data = fake)


    if(donnee==2):
        plt.title('Graphique du tableau préliminaire')
        plt.ylim(0,1.1)

        barWidth = 0.25
        bars1 = [0.5877,0.6707,0.9996,1.0000,0.8928,1.00]
        bars2 = [0.63,0.62,0.65,0.57,0.87,0.92]
        bars3 = [0.5757,0.6673,0.6054,0.5665,0.7727,0.9366]


# Set position of bar on X axis
        r1 = np.arange(len(bars1))
        r2 = [x + barWidth for x in r1]
        r3 = [x + barWidth for x in r2]

# Make the plot
        plt.bar(r1, bars1, color='#7f6d5f', width=barWidth,
edgecolor='white', label='Train')
        plt.bar(r2, bars2, color='#557f2d', width=barWidth,
edgecolor='white', label='CV(+/- 0,01)')
        plt.bar(r3, bars3, color='#2d7f5e', width=barWidth,
edgecolor='white', label='Valid(ation)')

# Add xticks on the middle of the group bars
        plt.xlabel('Méthode de classification', fontweight='bold')
        plt.ylabel('Score', fontweight='bold')
        plt.xticks([r + barWidth for r in range(len(bars1))],
['NaiveBayes', 'SGDC', 'RandomForest', 'DecisionTree', 'QDA','MPL'])

# Create legend & Show graphic
        plt.legend()


    if(donnee==3):
        plt.title('Score CV avec MLP et differents preprocessing')
        plt.ylim(0,1.1)

        barWidth = 0.25
        bars1 = [0.9,0.85,0.76,0.64]
        bars2 = [0.9,0.85,0.78,0.64]
        bars3 = [0.9]


# Set position of bar on X axis
        r1 = np.arange(len(bars1))
        r2 = [x + barWidth for x in r1]
        r3 = [x + barWidth for x in r2]

# Make the plot
        plt.bar(r1, bars1, color='#7f6d5f', width=barWidth,
edgecolor='white', label='PCA')
        plt.bar(r2, bars2, color='#557f2d', width=barWidth,
edgecolor='white', label='SelectKBest')
        plt.bar(r3, bars3, color='#2d7f5e', width=barWidth,
edgecolor='white', label='PCA + SelectKBest')

# Add xticks on the middle of the group bars
        plt.xlabel('Méthode de classification', fontweight='bold')
        plt.ylabel('Score', fontweight='bold')
        plt.xticks([r + barWidth for r in range(len(bars1))],
['components = 140', 'components = 100', 'components = 60',
'components = 20', 'components = PCA = 170 & SelectKbest = 140'])

# Create legend & Show graphic
```

```
        plt.legend()

    plt.show()
    plt.close()


def duTP (data):
    plt.figure(figsize = (16,16))
    data.boxplot()
    plt.show()
    plt.close()

    plt.figure(figsize = (16,16))
    corr_mat = data.corr(method='pearson')
    a = sns.heatmap(corr_mat, annot=True, center=0)
    plt.show()
    plt.close()

    plt.figure(figsize = (16,16))
    data_num = data.copy()  # If you don't use "copy", any change in 
data_num will also result in a change in data
    data_num['target']= data_num['target'].astype('category')
    data_num['target'] = data_num['target'].cat.codes
    b = sns.heatmap(data_num)
    plt.show()
    plt.close()



if __name__=="__main__":
    sns.set(font_scale=1.4,style="whitegrid") #set styling 
preferences


    url = "C:\\Users\\isabe\\Downloads\\monet-
master\\starting_kit\\c1_input_data\\perso_train.data"
    data = data_io.read_as_df('C:\\Users\\isabe\\Downloads\\monet-
master\\starting_kit\\c1_input_data\\perso') #tout
    df = pd.read_csv(url ,delimiter=' ') # toutes les données sans 
le head, ni les labels
    labels = data.iloc[1:,-1] # label : true ou false?



    pca=PCA(n_components=200)
    x_reduced=pca.fit_transform(data)
   # print(data)
    x_reduced = pd.DataFrame(x_reduced)
    #print (x_reduced.shape)
    #data = x_reduced
```

```
funcPCA (data)
funcDiagbaton(1)
funcDiagbaton(2)
funcDiagbaton(3)
duTP(data)
```