Devoir de Machine Learning 2

Exercice 1

1) Répartition des données par classe

```
Entrée [10]: df['classe'].value_counts()

Out[10]:  1    6282
         -1    2243
         Name: classe, dtype: int64
```

Nous avons **6282** individus dans la classe **1** et **2243** dans la classe **-1.** On est dans un problème de **classification**.

2) a) Proposition d'arbre de décision

- D'abord après avoir indexé la variable cible et les descripteurs nous scindons nos données en 80% pour l'apprentissage et 20% pour le test.

```
Spécifier la proportion de données pour l'apprentissage (entre 0 et 1) : 0.8
==========================
Proportion des données d'apprentissage :  6820
Proportion des données de test :  1705
```

- Ensuite nous utilisons une méthode de recherche de paramètres optimaux qui nous donne un score de 92% avec entropie pour critère de sélection et 8 comme profondeur maximale.
- Apres construction du modèle avec ces paramètres on obtient les performances ci-dessous :

```
Apprentissage en cours ...
Evaluation de performance ...
=======Matrice de confusion ============
[[ 388   71]
 [  83 1163]]
=======Autres indicateurs de performance ==========
              precision    recall  f1-score   support

          -1       0.82      0.85      0.83       459
           1       0.94      0.93      0.94      1246

    accuracy                           0.91      1705
   macro avg       0.88      0.89      0.89      1705
weighted avg       0.91      0.91      0.91      1705
```

- On a 71 faux négatifs et 83 faux positifs avec **82% des éléments de la classe -1** correctement prédit **et 94% de la classe +1** correctement prédit. Sur les 1705 individus réels 91% a été correctement prédit pour le modèle ce qui est acceptable.
- A présent nous allons appliquer une validation croisée avec cv=10

```
Spécifier la valeur de CV : (defaut = 5)
10
========================
Moyenne des scores : 0.918
variance des scores : 0.00014
Score par étape de validation : [0.89929742 0.90046838 0.92497069 0.92018779 0.90962441 0.914
31925
 0.92957746 0.93896714 0.91549296 0.92488263]
```

On obtient un score moyen de 91,8% ce qui n'est pas loin de notre score obtenu avec les paramètres optimaux par conséquent nous pouvant dire que le modèle est bon.

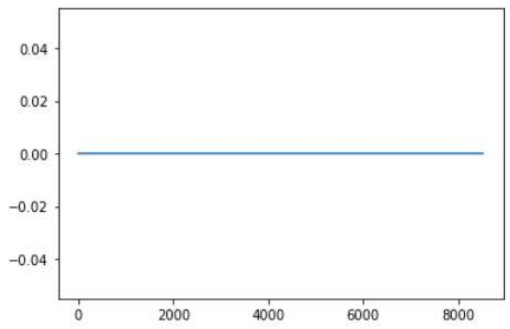
b) On a 49 attributs qui ont été utilisé pour construire le modèle.

Liste des 11 meilleurs attributs

	Variable	Importance
100	test_area_distribution_mean10	0.336808
98	test_area_distribution_mean8	0.109476
103	test_area_distribution_mean13	0.101874
96	test_area_distribution_mean6	0.101671
14	spectral_flux_variance	0.074142
89	frame_diff_distribution_32	0.049421
10	spectral_centroid_variance	0.028437
90	frame_diff_distribution_33	0.021257
91	test_area_distribution_mean1	0.019258
5	short_time_energy_mean	0.016065
15	fondamental_frequency_mean	0.011296

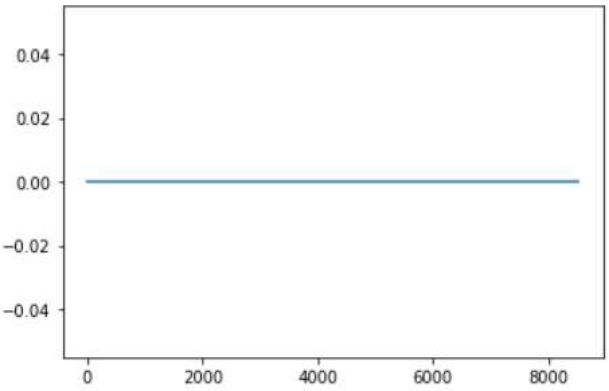
- Représentation graphique de frame_diff_distribution_30

```
Entrée [58]: df['frame_diff_distribution_30'].plot()
Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x2569a621448>
```



- Représentation graphique de frame_diff_distribution_31

```
Entrée [59]: df['frame_diff_distribution_31'].plot()

Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x256ccecee08>
```



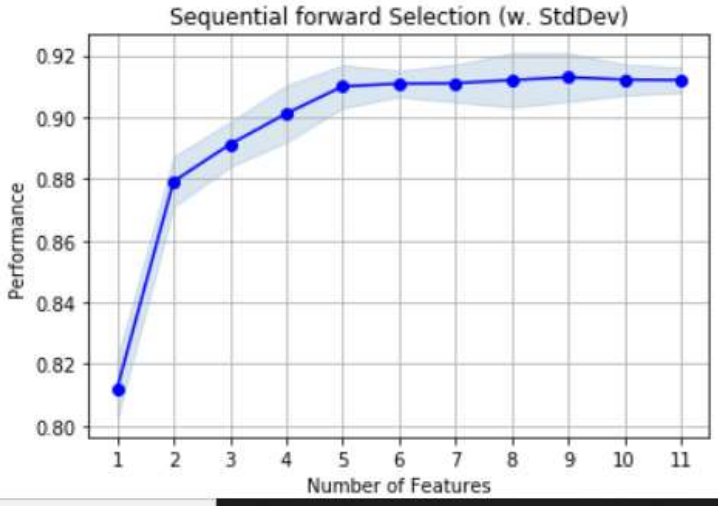
- Les deux figurent montres que les deux variables ont une valeur constante et égale à 0 ce qui signifie que les deux n'apportent pas d'information raison pour laquelle elles n'ont pas été sélectionnées.

c) Apprentissage avec les 11 premiers meilleurs attributs

```
Apprentissage en cours ...
Evaluation de performance ...
=======Matrice de confusion ============
[[ 360   60]
 [  70 1215]]
=======Autres indicateurs de performance ==========
              precision    recall  f1-score   support

          -1       0.84      0.86      0.85       420
           1       0.95      0.95      0.95      1285

    accuracy                           0.92      1705
   macro avg       0.90      0.90      0.90      1705
weighted avg       0.92      0.92      0.92      1705
```

- Sélection de variables avec la méthode forward avec valeur de cv égale à 5

```
Choisissez la méthode de sélection de variables :
0 : Backward selection
1 : Forward selection
 (defaut =Forward selection)

spécifier la valeur du CV (cross-validation)
 (defaut = 5)
```



- L'importance des attributs se résume dans la figure ci-dessous :

	feature_names	avg_score
1	(test_area_distribution_mean13,)	0.812174
2	(test_area_distribution_mean10, test_area_dist...	0.879176
3	(test_area_distribution_mean6, test_area_distr...	0.891199
4	(test_area_distribution_mean6, test_area_distr...	0.901174
5	(frame_diff_distribution_32, test_area_distrib...	0.909971
6	(short_time_energy_mean, frame_diff_distributi...	0.910851
7	(short_time_energy_mean, fondamental_frequency...	0.910995
8	(short_time_energy_mean, fondamental_frequency...	0.912021
9	(short_time_energy_mean, fondamental_frequency...	0.912903
10	(short_time_energy_mean, spectral_flux_varianc...	0.912167
11	(short_time_energy_mean, spectral_centroid_var...	0.912022

- Avec cette méthode on a le nombre optimale d'attribut qui égale à 9 car sur le graphe on remarque que avec 9 attribut on obtient la meilleure performance de 0.912903

3) Les 6 meilleurs attributs sélectionnés

```
======================
Attributs :  ['short_time_energy_mean', 'frame_diff_distribution_32', 'test_area_distributi
on_mean8', 'test_area_distribution_mean10', 'test_area_distribution_mean13', 'test_area_dis
tribution_variance6']
Variable cible :  classe
Repartition des individus par classe
 1    6282
-1    2243
Name: classe, dtype: int64
```

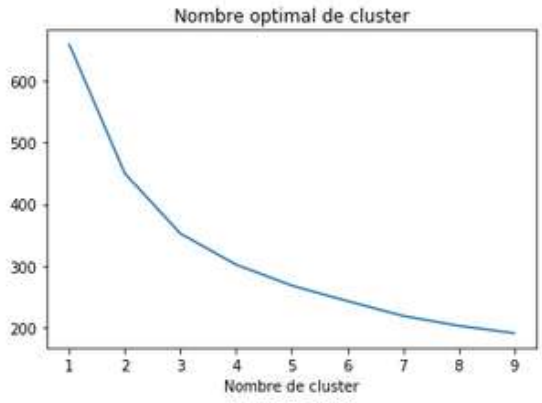
- Indication de performances par modèles avec les 06 meilleurs attributs

Modèle	**Accuracy**	**Validation croisée**	
		Moyenne	**Variances**
CART	0.92	0.912	0.00007
Random Forest	0.93	0.924	0.00005
SVM	0.83	0.840	0.00003
Boosting	0.93	0.925	0.00000
Bagging	0.93	0.922	0.00004

Le tableau résume les performances de l'exécution de chaque modèle avec les paramètres par défaut sur les données suivi de la validation croisés. Ainsi on retient que la méthode Boosting a la variance la plus faible donc on peut dire que cette méthode est meilleure que CART.

4) a) Nous avons essayé de construire des clusters allant de 1 à 10 et on obtient le graphe suivant :

```
spécifier les valeurs de K : (séparer les valeurs par une virgule ex. 1,7,4,5,8)
Pour sélectionner plusieurs, utiliser 'debut:fin:pas' (ex. 1:20:1)1:10:1
```



On remarque que le graphe est décroisant. Cependant on a un nœud au point 2 où la courbe change d'allure et continue au point 3 où elle fait un deuxième nœud avant de continuer au point 9. Ainsi nous concluons que le nombre optimal de cluster est de 2 .

b)Génération des clusters et ajout du label au data frame

```
Spécifier le nombre de cluster (K) que vous souhaiteriez avoir
(defaut=4)
2
Sélectionner l'approche d'initialisation des centroids :
0 : k-means++
1 : random
 (par défaut =k-means++)
0
```

III.2. Insertion des classes dans la dataframe

```
crée [103]: df["kmeans_class"] = labels

crée [104]: df.columns

[104]: Index(['shot_length', 'motion_dist_mean', 'motion_dist_variance',
       'frame_difference_dist_mean', 'frame_difference_dist_variance',
       'short_time_energy_mean', 'short_time_energy_variance', 'ZCR_mean',
       'ZCR_variance', 'spectral_centroid_mean',
       ...
       'test_area_distribution_variance9', 'test_area_distribution_variance10',
       'test_area_distribution_variance11',
       'test_area_distribution_variance12',
       'test_area_distribution_variance13',
       'test_area_distribution_variance14',
       'test_area_distribution_variance15',
       'test_area_distribution_variance16', 'classe', 'kmeans_class'],
      dtype='object', length=124)
```

On a deux cluster labelisés avec 0 et 1.

c) Sélection de la cible et des descripteurs

```
Sélectionner les indices des attributs (séparer par une virgule ex. 1,4,5,8)
Pour sélectionner plusieurs, utiliser ':' (ex. 0:7): 5,89,111,98,100,103
======================
Attributs :  ['short_time_energy_mean', 'frame_diff_distribution_32', 'test_area_distribution_mean8', 'test_ar
ea_distribution_mean10', 'test_area_distribution_mean13', 'test_area_distribution_variance6']
Variable cible :  kmeans_class
Repartition des individus par classe
1    5311
0    3214
Name: kmeans_class, dtype: int64
```

- Instanciation de la méthode choisie pour l'apprentissage(Boosting)

```
Choisir l'algorithme de machine learning :
0 : CART
1 : Foret aleatoire
2 : Bagging
3 : Boosting)
4 : SVM
3
Sélection le type de prédiction :
0 : classification
1 : régression
NB : laisser ce champ vide si non définie (par defaut = Classification)

Spécifier le nombre d'arbre :
NB : laisser ce champ vide si non définie (par defaut = 50)

==========================
Méthode de ML :  Boosting - Classifier
Nombre d'arbres : 50
```

- Apres apprentissage on a les performances suivantes :

```
Apprentissage en cours ...
Evaluation de performance ...
=======Matrice de confusion ============
[[ 617    8]
 [  14 1066]]
=======Autres indicateurs de performance ==========
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       625
           1       0.99      0.99      0.99      1080

    accuracy                           0.99      1705
   macro avg       0.99      0.99      0.99      1705
weighted avg       0.99      0.99      0.99      1705
```

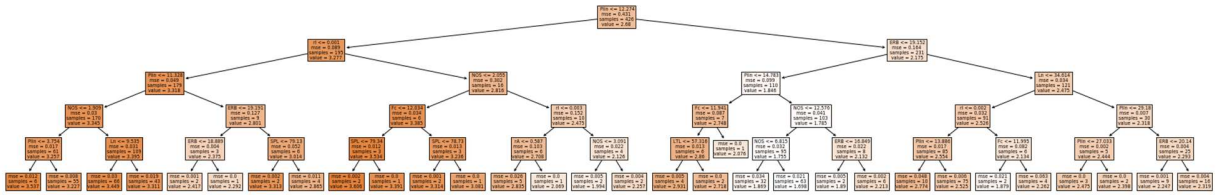
On a une précision de 98% pour la classe 0 et 99% pour la classe 1 d'où une moyenne de 99% contre 93% ce qui signifie que le modèle s'est nettement amélioré.

Exercice 2

- Nous utilisons la méthode CART pour faire la régression a ce niveau. Nous allons chercher les paramètres optimaux pour la construction du modèle. Les meilleurs paramètres sont mse comme critère et 5 comme max_depth. Après la construction du modèle on obtient les performances suivantes :

  coefficient de détermination(R^2) : 0.8767584920545115
  erreur quadratique moyenne (EQM) : 0.055367438579659235

- Représentation de l'arbre de décision



- Code de l'arbre généré correspond aux règles de décisions obtenue avec construction de l'arbre. Les données sont divisées en deux groupes suivant que la variable Plin >=12,27 ou Plin<=12,27

```
|---  Plin <= 12.27
|    |---  rl <= 0.00
|    |    |---  Plin <= 11.33
|    |    |    |---  NOS <= 1.91
|    |    |    |    |---  Plin <= 3.75
|    |    |    |    |    |--- value: [3.54]
|    |    |    |    |---  Plin >  3.75
|    |    |    |    |    |--- value: [3.23]
|    |    |    |---  NOS >  1.91
|    |    |    |    |---  Ln <= 9.53
|    |    |    |    |    |--- value: [3.45]
|    |    |    |    |---  Ln >  9.53
|    |    |    |    |    |--- value: [3.31]
|    |    |---  Plin >  11.33
|    |    |    |---  Plin <= 11.53
|    |    |    |    |---  ERB <= 18.89
|    |    |    |    |    |--- value: [2.42]
|    |    |    |    |---  ERB >  18.89
|    |    |    |    |    |--- value: [2.29]
|    |    |    |---  Plin >  11.53
|    |    |    |    |---  Ln <= 17.00
|    |    |    |    |    |--- value: [2.87]
|    |    |    |    |---  Ln >  17.00
|    |    |    |    |    |--- value: [3.31]
|    |---  rl >  0.00
|    |    |---  NOS <= 2.06
|    |    |    |---  Plin <= 5.46
|    |    |    |    |---  LTL <= 47.08
|    |    |    |    |    |--- value: [3.61]
|    |    |    |    |---  LTL >  47.08
|    |    |    |    |    |--- value: [3.39]
```

```
|   |   |   |---  Plin >  5.46
|   |   |   |   |---  ERB <= 22.24
|   |   |   |   |   |--- value: [3.31]
|   |   |   |   |---  ERB >  22.24
|   |   |   |   |   |--- value: [3.08]
|   |   |---  NOS >  2.06
|   |   |   |---  rl <= 0.00
|   |   |   |   |---  rA <= 0.59
|   |   |   |   |   |--- value: [2.84]
|   |   |   |   |---  rA >  0.59
|   |   |   |   |   |--- value: [2.07]
|   |   |   |---  rl >  0.00
|   |   |   |   |---  NOS <= 3.09
|   |   |   |   |   |--- value: [1.99]
|   |   |   |   |---  NOS >  3.09
|   |   |   |   |   |--- value: [2.26]
|---  Plin >  12.27
|   |---  ERB <= 19.15
|   |   |---  Plin <= 14.78
|   |   |   |---  Fc <= 11.94
|   |   |   |   |---  Ln <= 20.60
|   |   |   |   |   |--- value: [2.93]
|   |   |   |   |---  Ln >  20.60
|   |   |   |   |   |--- value: [2.72]
|   |   |   |---  Fc >  11.94
|   |   |   |   |--- value: [2.08]
|   |   |---  Plin >  14.78
|   |   |   |---  NOS <= 12.58
|   |   |   |   |---  NOS <= 6.81
|   |   |   |   |   |--- value: [1.87]
|   |   |   |   |---  NOS >  6.81
|   |   |   |   |   |--- value: [1.70]
|   |   |   |---  NOS >  12.58
|   |   |   |   |---  ERB <= 16.85
|   |   |   |   |   |--- value: [1.89]
|   |   |   |   |---  ERB >  16.85
|   |   |   |   |   |--- value: [2.21]
|   |---  ERB >  19.15
|   |   |---  Ln <= 34.61
|   |   |   |---  rl <= 0.00
|   |   |   |   |---  Plin <= 13.89
|   |   |   |   |   |--- value: [2.77]
|   |   |   |   |---  Plin >  13.89
|   |   |   |   |   |--- value: [2.52]
|   |   |   |---  rl >  0.00
|   |   |   |   |---  Fc <= 11.99
|   |   |   |   |   |--- value: [1.88]
|   |   |   |   |---  Fc >  11.99
|   |   |   |   |   |--- value: [2.26]
|   |   |---  Ln >  34.61
|   |   |   |---  Plin <= 29.18
|   |   |   |   |---  NOS <= 10.47
|   |   |   |   |   |--- value: [2.47]
|   |   |   |   |---  NOS >  10.47
|   |   |   |   |   |--- value: [2.40]
|   |   |   |---  Plin >  29.18
|   |   |   |   |---  ERB <= 20.14
|   |   |   |   |   |--- value: [2.25]
|   |   |   |   |---  ERB >  20.14
|   |   |   |   |   |--- value: [2.32]
```