PYTHON FOR DATA ANALYSIS
AVILA DATASET

Analyse du dataset Avila

Par Antoine WILLE et Tanguy WALRAVE

SOMMAIRE

PRÉSENTATION DU DATASET

ANALYSE DU DATASET

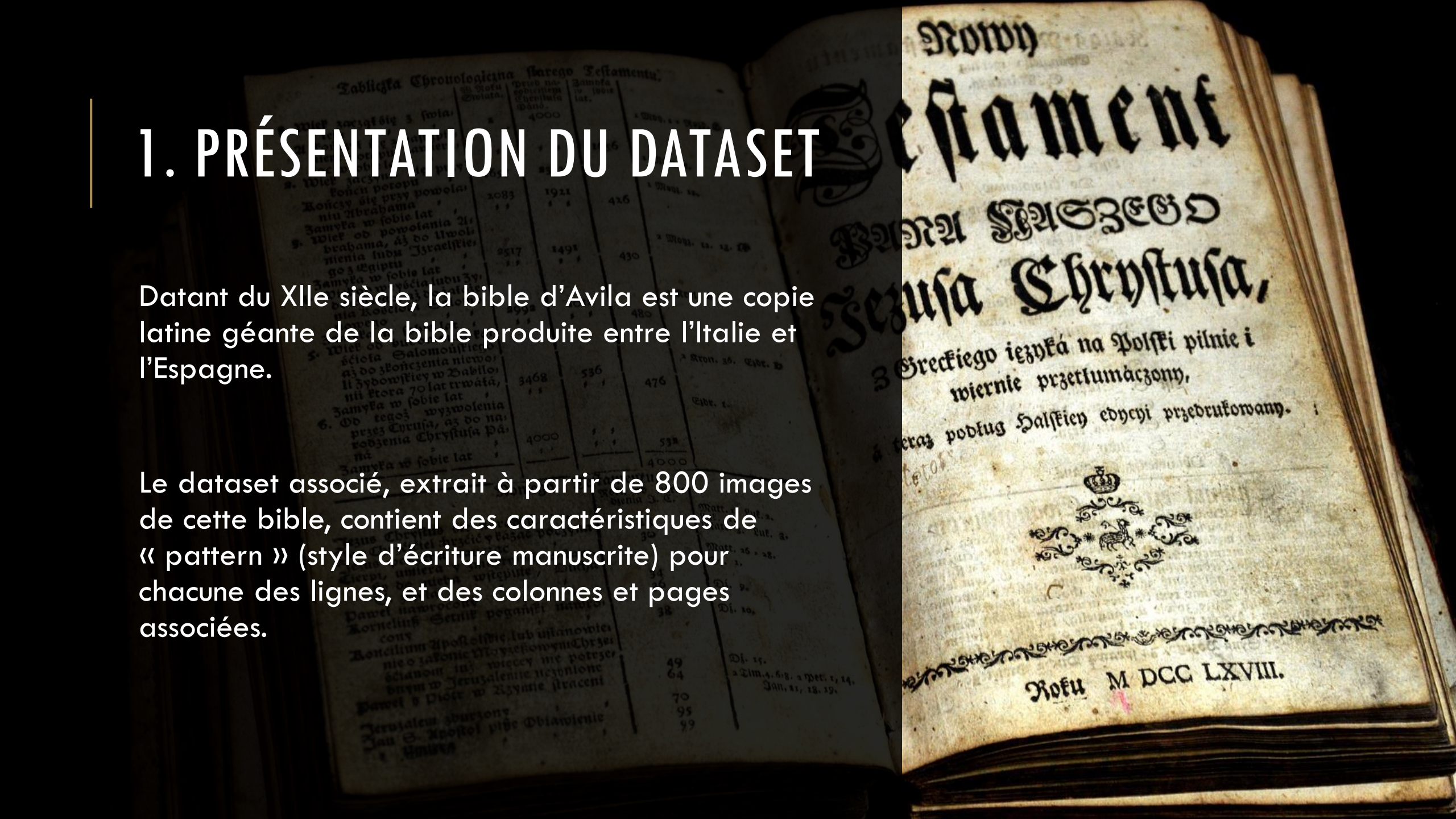
PRÉDICTION ET MODÈLES

API DJANGO

1. PRÉSENTATION DU DATASET

Datant du XIIe siècle, la bible d'Avila est une copie latine géante de la bible produite entre l'Italie et l'Espagne.

Le dataset associé, extrait à partir de 800 images de cette bible, contient des caractéristiques de « pattern » (style d'écriture manuscrite) pour chacune des lignes, et des colonnes et pages associées.



1. PRÉSENTATION DU DATASET

Le dataset, présenté sous forme de tableau nous fournit donc les caractéristiques suivantes :

- F1 : intercolumnar distance (La distance entre les deux colonnes)
- F2 : upper margin (La marge supérieure)
- F3 : lower margin (La marge inférieure)
- F4 : exploitation (La quantité d'encre sur la colonne)
- F5 : row number (Le nombre de lignes de la colonne)
- F6 : modular ratio (Le ratio entre la hauteur et la largeur des caractères)
- F7 : interlinear spacing (L'espacement entre les lignes)
- F8 : weight (La quantité d'encre sur la ligne)
- F9 : peak number (Le nombre de pics verticaux)
- F10 : modular ratio / interlinear spacing (Le ratio entre F6 et F7)

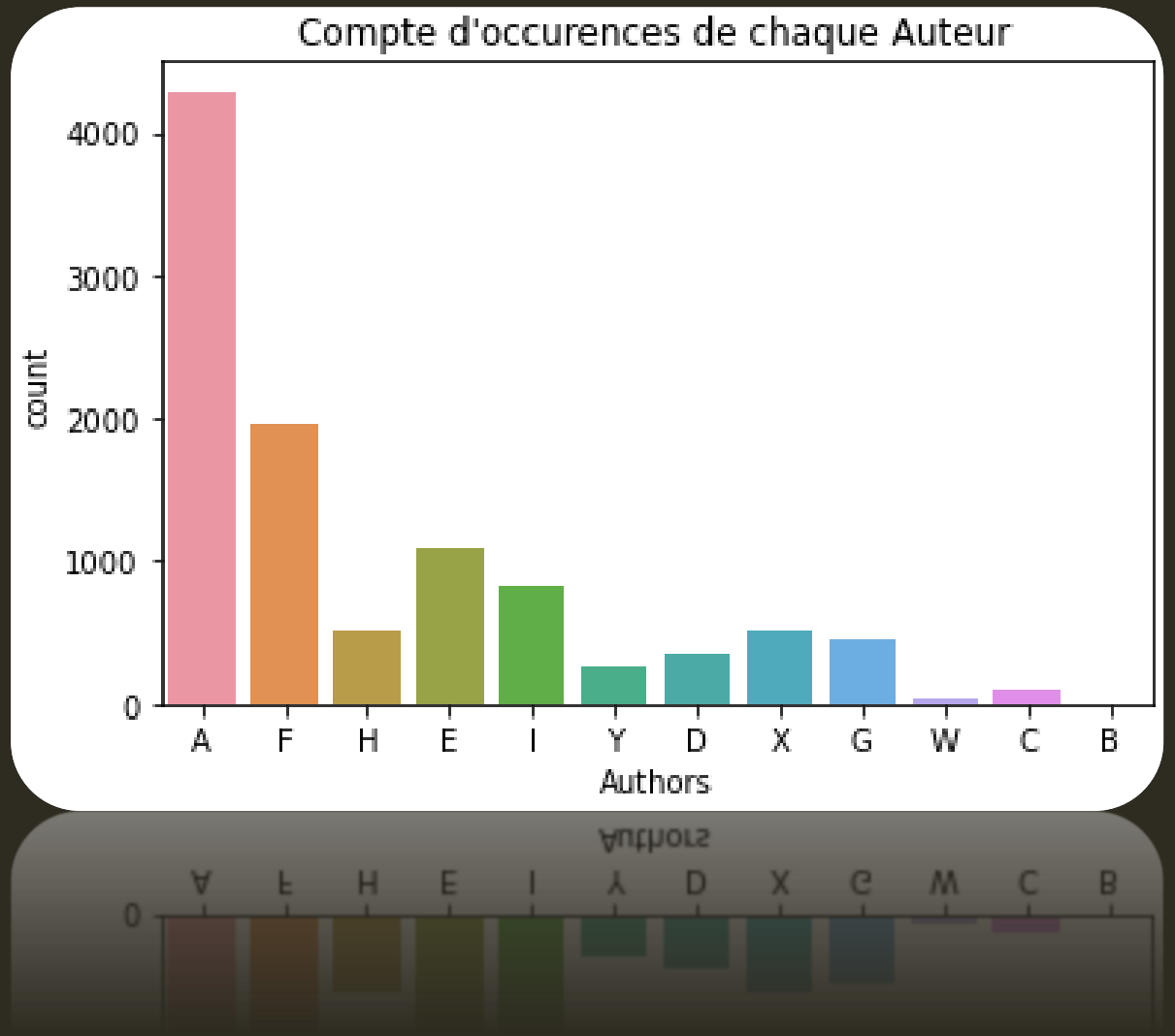
Les différentes « classes » sont des lettres distinguant les 12 auteurs (A, B, C, D, E, F, G, H, I, W, X, Y) et se trouvent à la fin de chaque échantillon.

Toutes ces données sont normalisées par une normalisation Z-score.

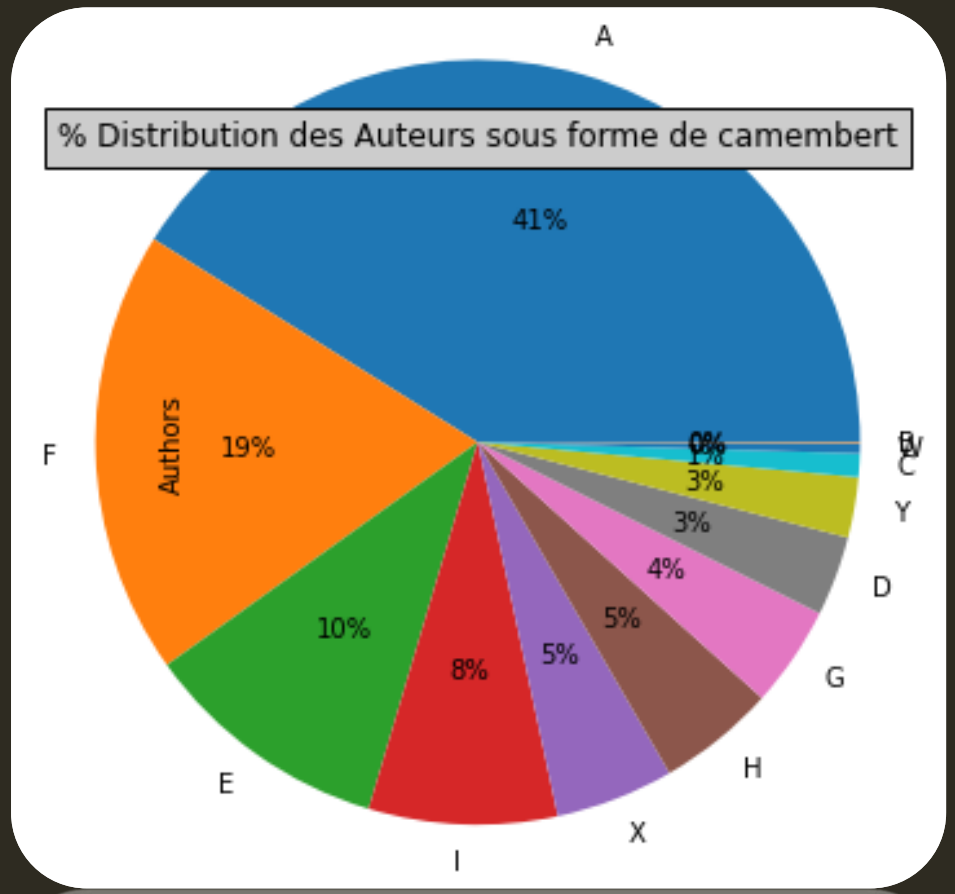
2. ANALYSE DU DATASET

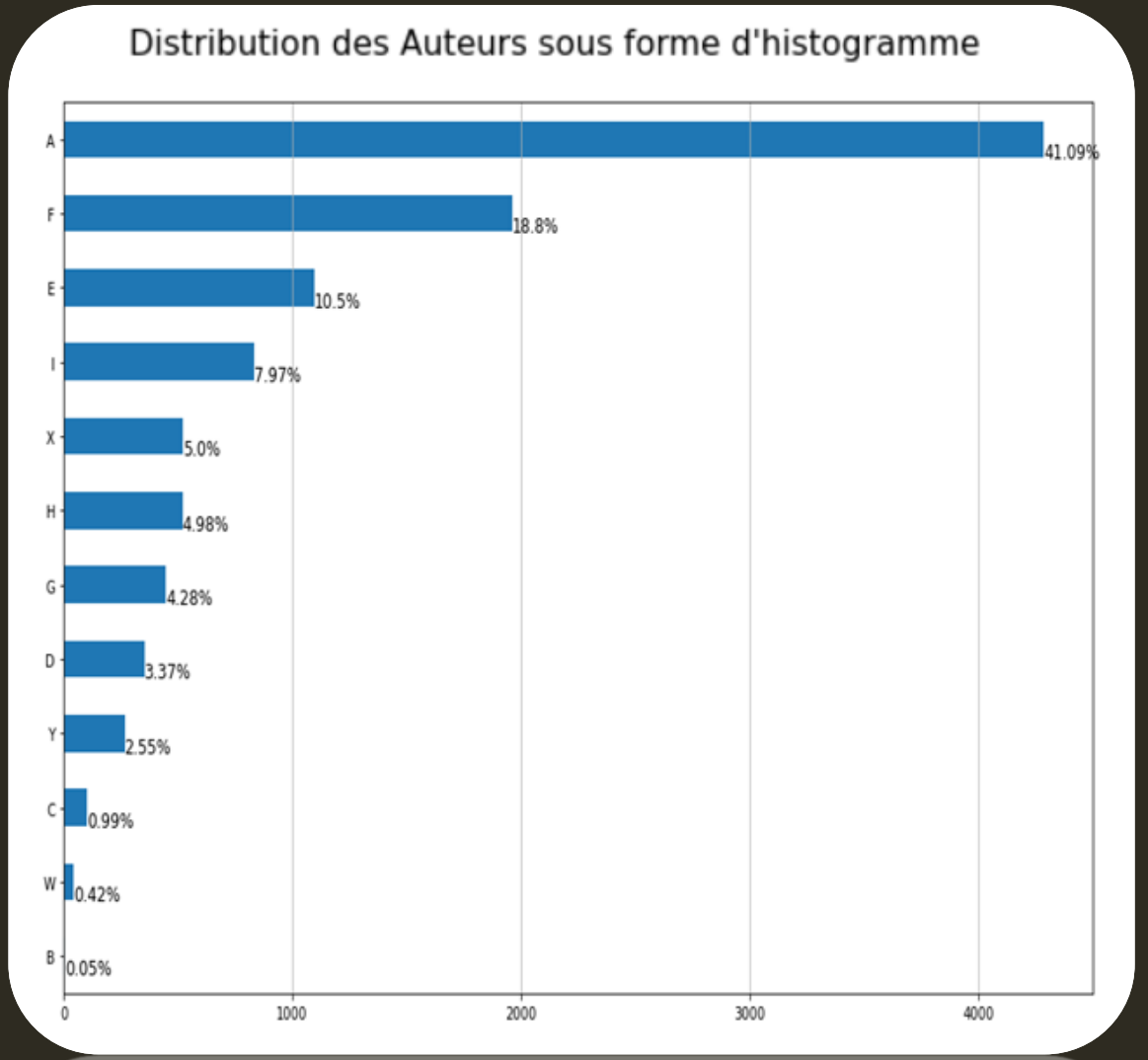
L'objectif final de ce projet est d'arriver à classifier chaque copiste en fonction de son style d'écriture (déterminée par les différentes caractéristiques) afin de pouvoir faire des prédictions à partir d'échantillons de valeurs de caractéristiques.

Nous avons commencé par analyser le dataset et les possibles corrélations entre les caractéristiques.



2. ANALYSE DU DATASET





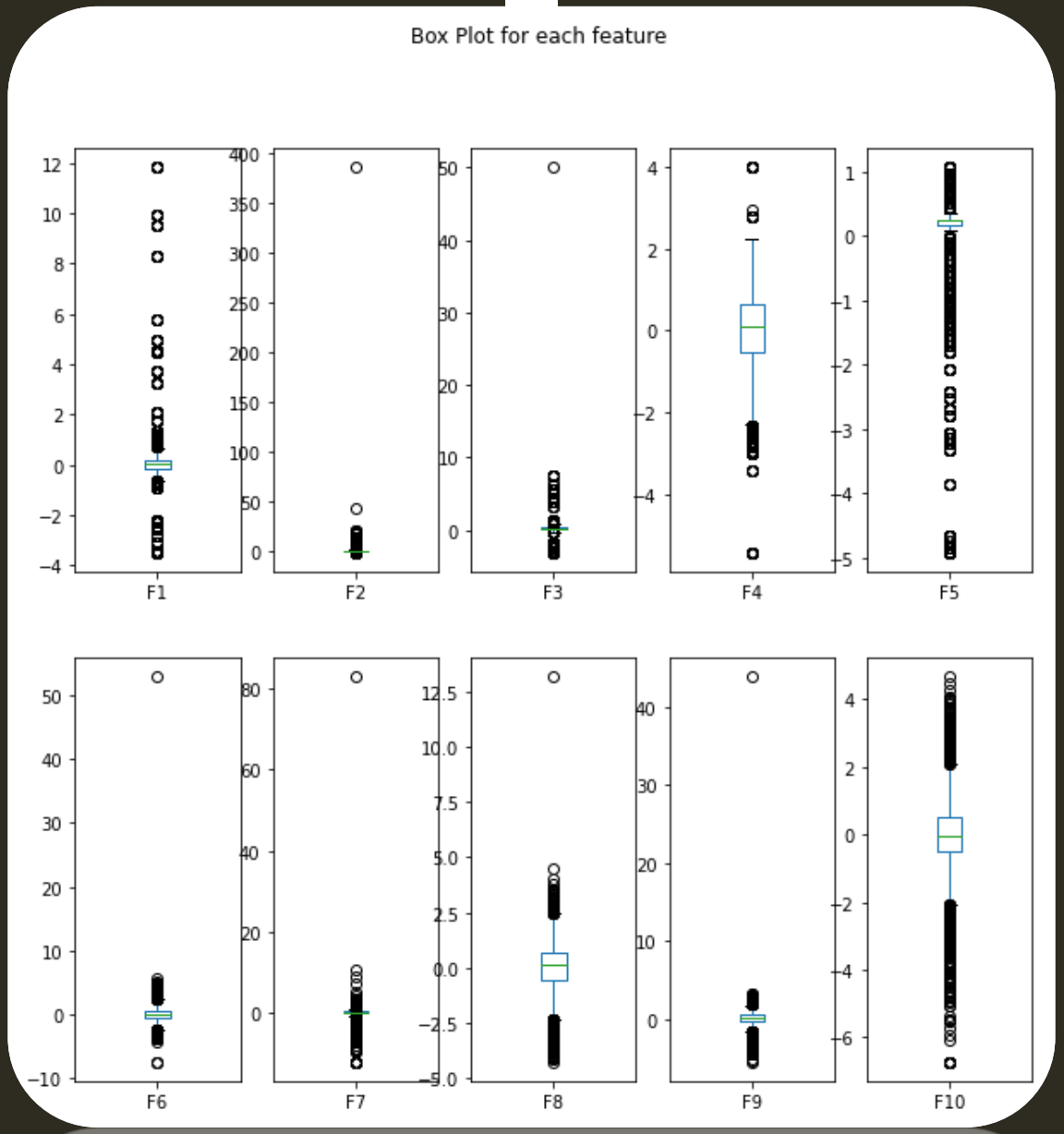
2. ANALYSE DU DATASET

Sur la figure 1 on peut observer quelques statistiques pour chaque feature: déviation standard, quartiles, valeurs moyennes, minimum et maximum

1

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
mean	0.000852	0.033611	-0.000525	-0.002387	0.006370	0.013973	0.005605	0.010323	0.012914	0.000818
std	0.991431	3.920868	1.120202	1.008527	0.992053	1.126245	1.313754	1.003507	1.087665	1.007094
mln	-3.498799	-2.426761	-3.210528	-5.440122	-4.922215	-7.450257	-11.935457	-4.247781	-5.486218	-6.719324
25%	-0.128929	-0.259834	0.064919	-0.528002	0.172340	-0.598658	-0.044076	-0.541992	-0.372457	-0.516097
50%	0.043885	-0.055704	0.217845	0.095763	0.261718	-0.058835	0.220177	0.111803	0.064084	-0.034513
75%	0.204355	0.203385	0.352988	0.658210	0.261718	0.564038	0.446679	0.654944	0.500624	0.530855
max	11.819916	386.000000	50.000000	3.987152	1.066121	53.000000	83.000000	13.173081	44.000000	4.671232

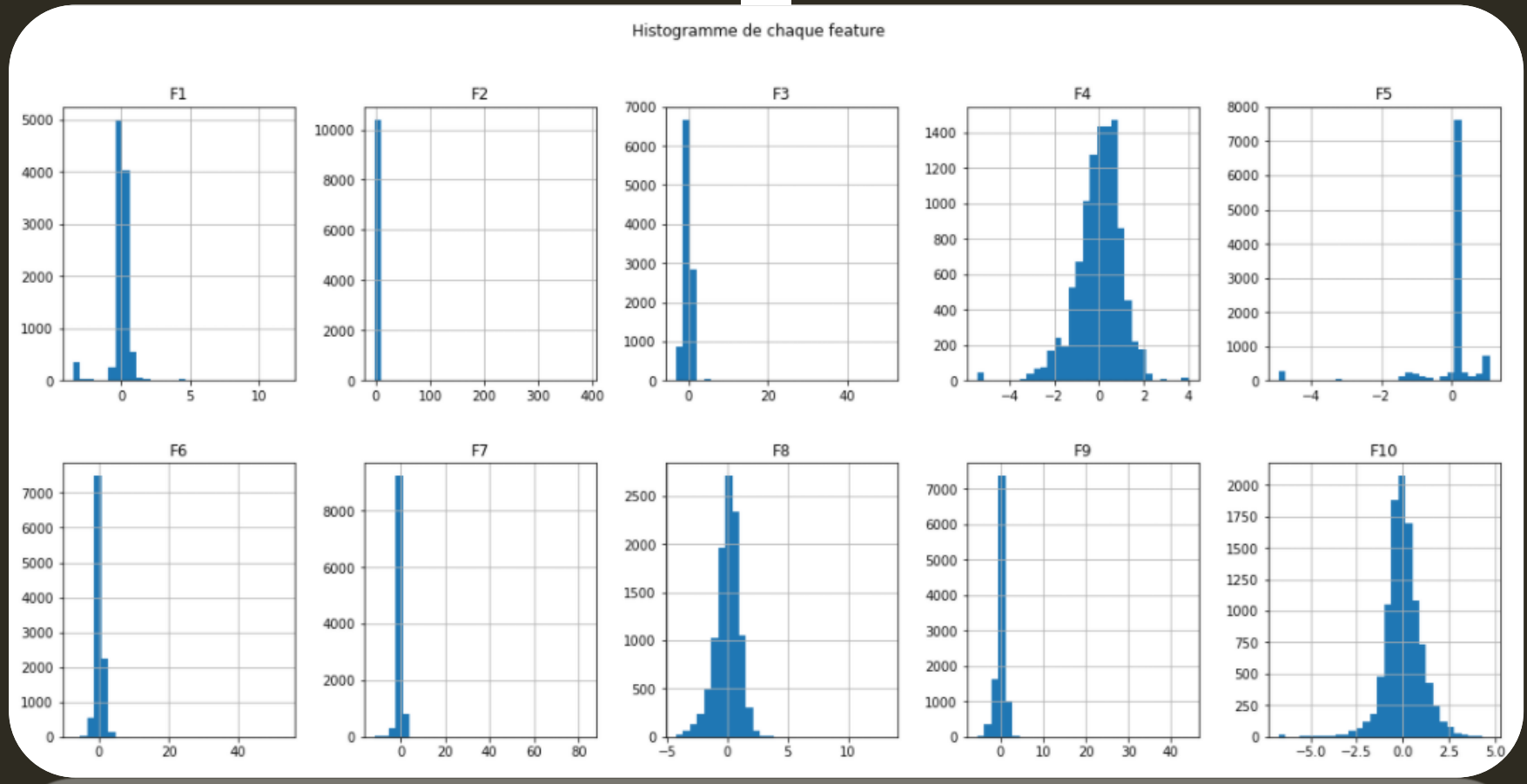
2



2. ANALYSE DU DATASET

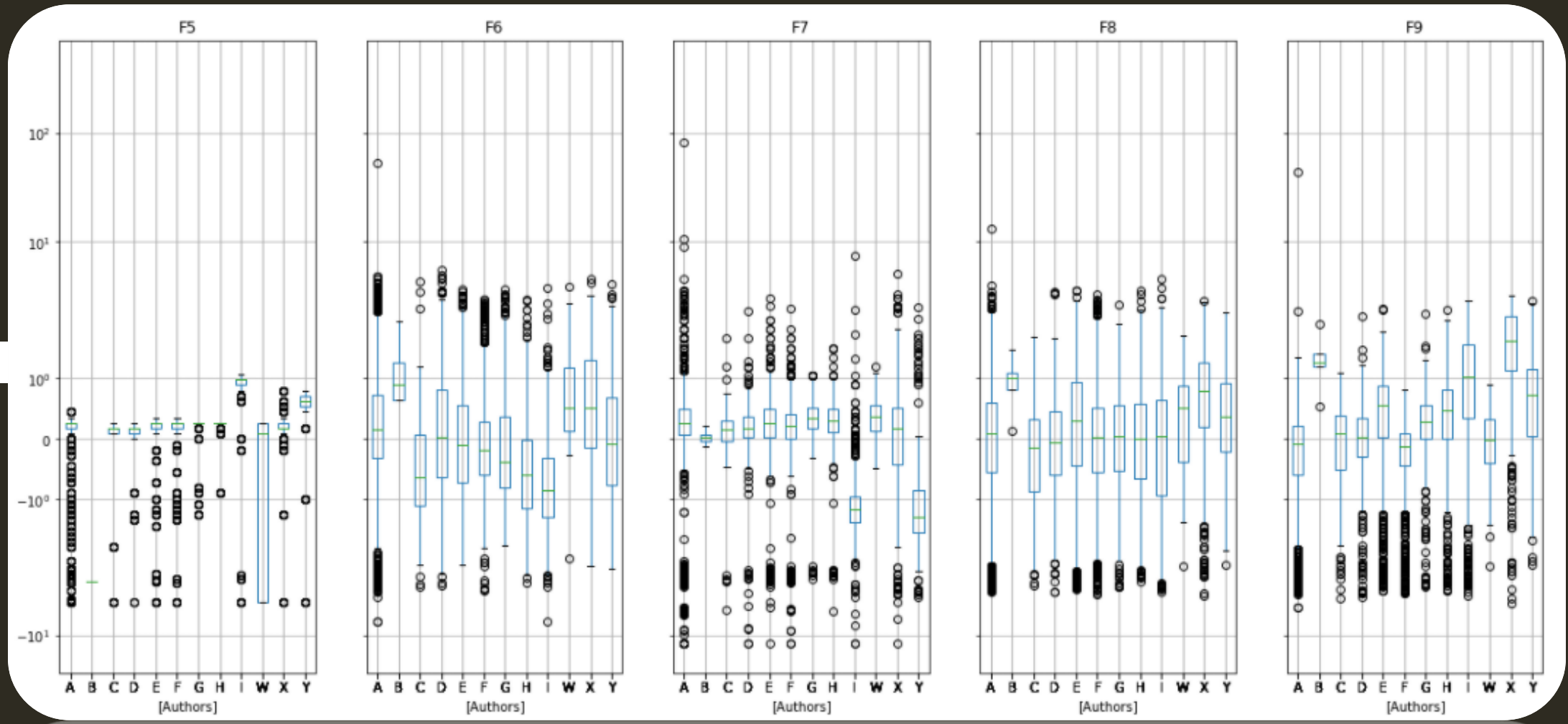
La distribution des features représentées par les figures 2 et 3 nous indiquent que certains semblent suivre des distributions gaussiennes (F4, F8, F10)

3



2. ANALYSE DU DATASET

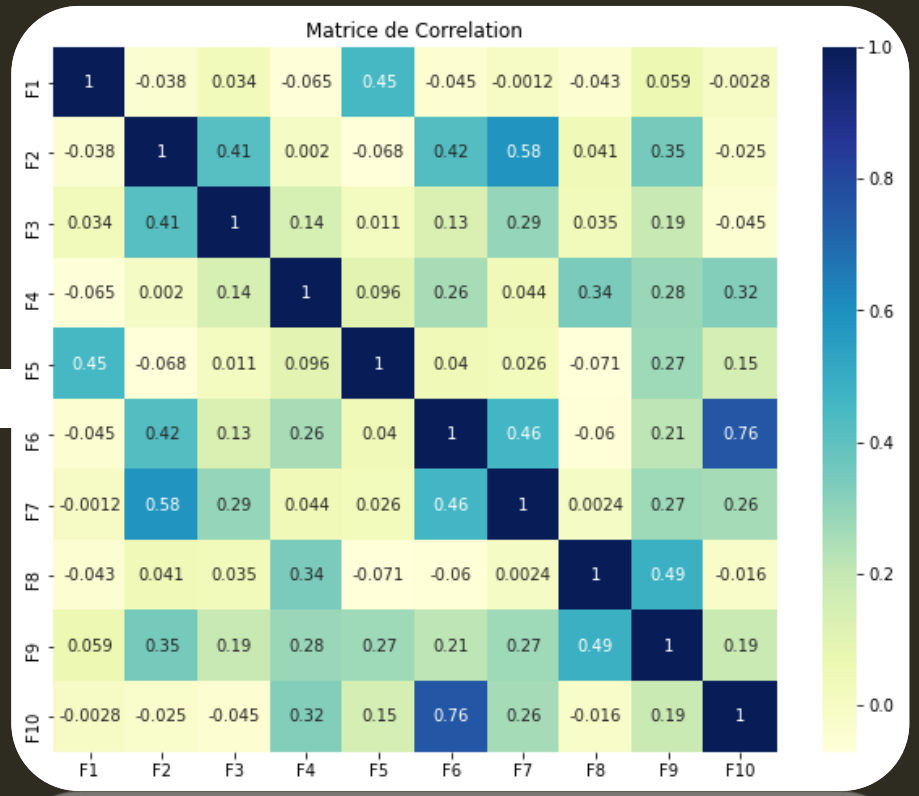
Distribution de chaque feature par auteur



2. ANALYSE DU DATASET

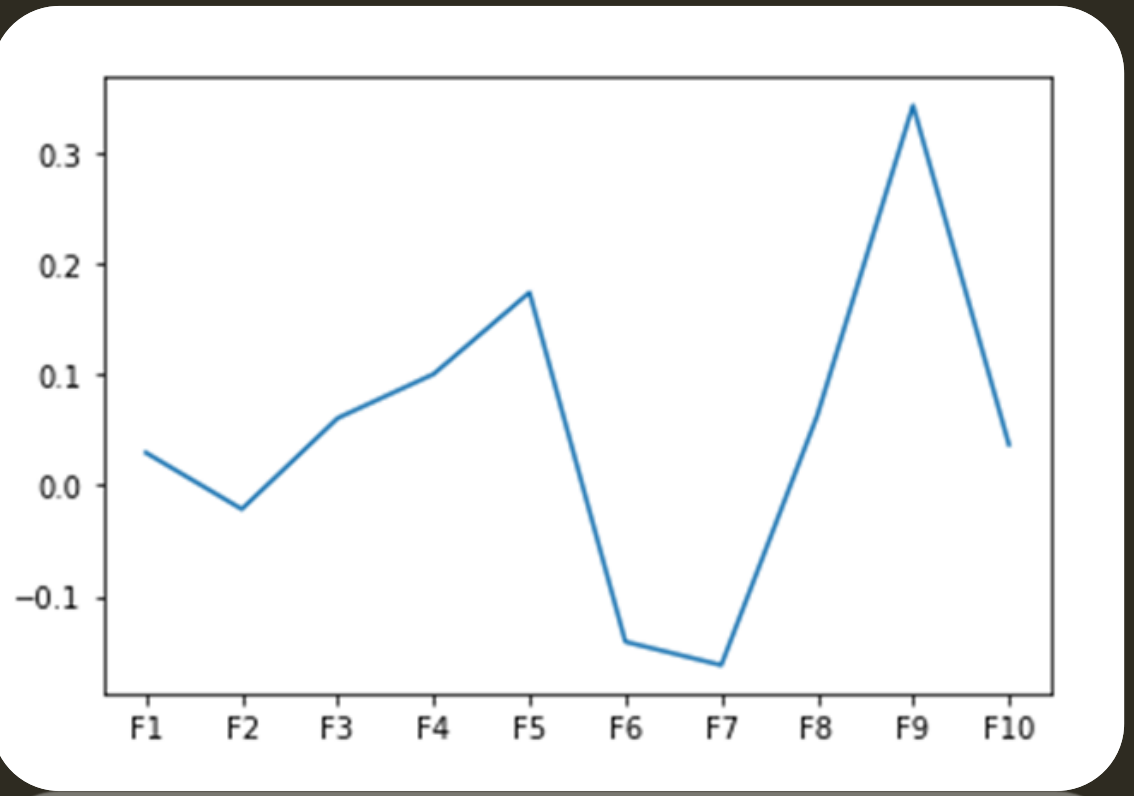
Corrélations entre toutes les données

5



Corrélations entre les features et auteurs

6



2. ANALYSE DU DATASET

En analysant ces différents graphiques, nous pouvons en tirer des informations intéressantes sur les relations existantes entre les features et les classes.

Par exemple, en analysant le graphique 6, nous pouvons observer que les caractéristiques F5 et F9 semblent être en forte corrélation avec la différenciation des auteurs.

Les features F6 et F7 quant à eux semblent avoir une très faibles corrélation, il pourrait même être possiblement envisageable de s'en débarrasser pour alléger le machine learning.

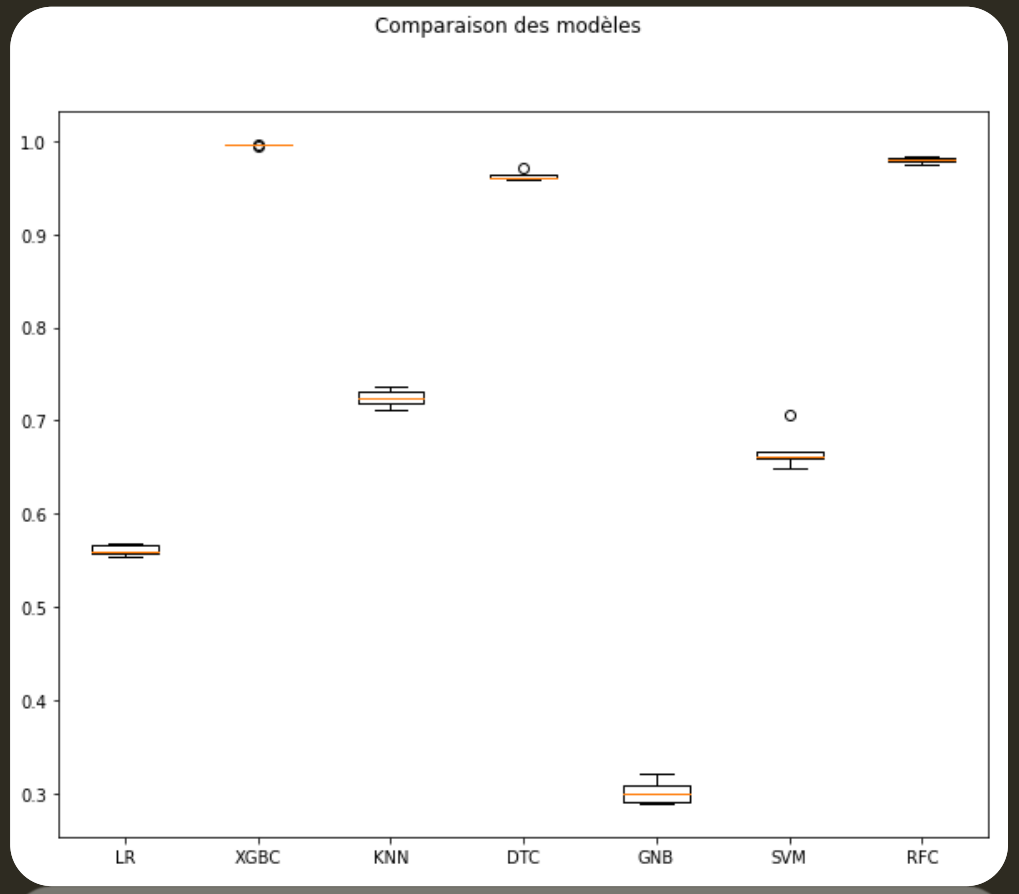
Nous pouvons également remarquer, en analysant le graphique 5, que les features 6 et 10 ont une forte corrélation entre elles, il pourrait être intéressant de n'en garder qu'un pour alléger le travail d'avantange sans perdre beaucoup d'informations.

3. PRÉDICTION ET MODÈLES

Pour réaliser ce projet, nous avons analysé 7 modèles de classification différents : Logistic regression, XGB, k-NN, Decision tree, Gaussian NB, SVM et Random Forest.

Nous avons commencé par les comparer via cross-validation en laissant les hyperparamètres par défaut

```
Model name:  LR
Results -->       Mean Accuracy:  0.5611697027804411   Standart Deviation Accuracy:  0.00522157192874021

Model name:  XGBC
Results -->       Mean Accuracy:  0.9955896452540747   Standart Deviation Accuracy:  0.0007670182166826273

Model name:  KNN
Results -->       Mean Accuracy:  0.7246404602109301   Standart Deviation Accuracy:  0.008847763754599699

Model name:  DTC
Results -->       Mean Accuracy:  0.9627037392138064   Standart Deviation Accuracy:  0.004418684035207255

Model name:  GNB
Results -->       Mean Accuracy:  0.3021093000958773   Standart Deviation Accuracy:  0.011965084684925732

Model name:  SVM
Results -->       Mean Accuracy:  0.6685522531160115   Standart Deviation Accuracy:  0.01987367363772737

Model name:  RFC
Results -->       Mean Accuracy:  0.9790987535953979   Standart Deviation Accuracy:  0.002869919376240814
```



3. PRÉDICTION ET MODÈLES

Il en ressort que 3 modèles s'avèrent réellement intéressants tant en terme de précision que d'exactitude pour notre usage : XGBC, DTC et RFC. Nous accordons également une mention honorable pour un 4è modèle: KNN.

Nous avons donc ensuite cherché à optimiser les hyperparamètres de ces 4 modèles cités ci-dessus.

Une courte commande nous permet de se rappeler des hyperparamètres possibles pour chacun :

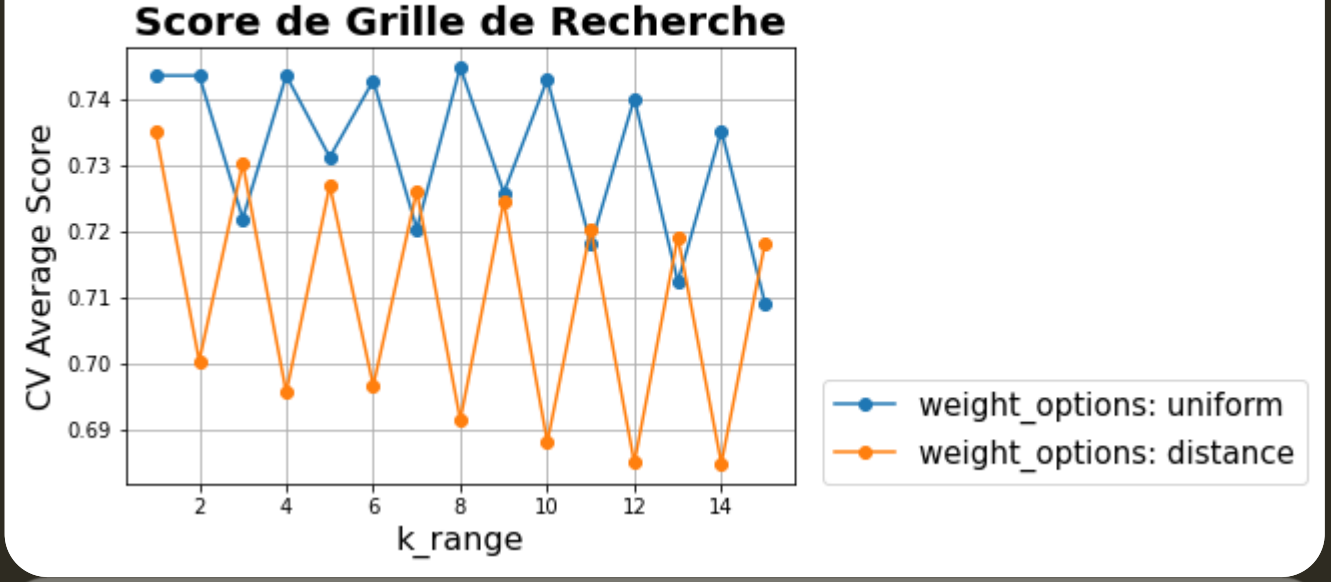
```
Model name: XGBC
--> Hyperparameters:
    - objective
    - use_label_encoder
    - base_score
    - booster
    - colsample_bylevel
    - colsample_bynode
    - colsample_bytree
    - gamma
    - gpu_id
    - importance_type
    - interaction_constraints
    - learning_rate
    - max_delta_step
    - max_depth
    - min_child_weight
    - missing
    - monotone_constraints
    - n_estimators
    - n_jobs
    - num_parallel_tree
    - random_state
    - reg_alpha
    - reg_lambda
    - scale_pos_weight
    - subsample
    - tree_method
    - validate_parameters
    - verbosity
    - eval_metric

Model name: KNN
--> Hyperparameters:
    - algorithm
    - leaf_size
    - metric
    - metric_params
    - n_jobs
    - n_neighbors
    - p
    - weights

Model name: DTC
--> Hyperparameters:
    - ccp_alpha
    - class_weight
    - criterion
    - max_depth
    - max_features
    - max_leaf_nodes
    - min_impurity_decrease
    - min_impurity_split
    - min_samples_leaf
    - min_samples_split
    - min_weight_fraction_leaf
    - presort
    - random_state
    - splitter

Model name: RFC
--> Hyperparameters:
    - bootstrap
    - ccp_alpha
    - class_weight
    - criterion
    - max_depth
    - max_features
    - max_leaf_nodes
    - max_samples
    - min_impurity_decrease
    - min_impurity_split
    - min_samples_leaf
    - min_samples_split
    - min_weight_fraction_leaf
    - n_estimators
    - n_jobs
    - oob_score
    - random_state
    - verbose
    - warm_start
```

3. PRÉDICTION ET MODÈLES

Pour chaque modèle choisi, nous avons fait varier certains de leurs paramètres et nous avons évalué ces variations sous formes de grilles de recherche. En voici les résultats :

Pour KNN :

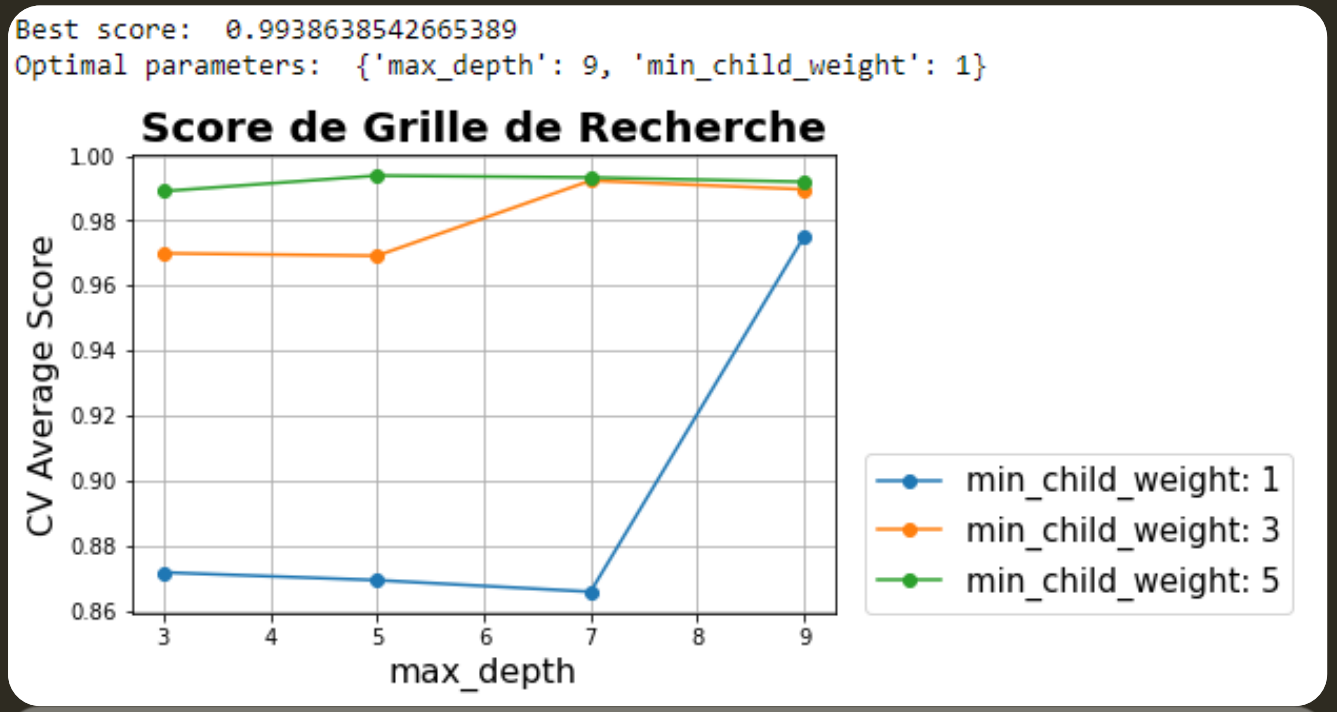
```
Best score:  0.7448705656759348
Optimal parameters:  {'n_neighbors': 4, 'weights': 'distance'}
```



3. PRÉDICTION ET MODÈLES

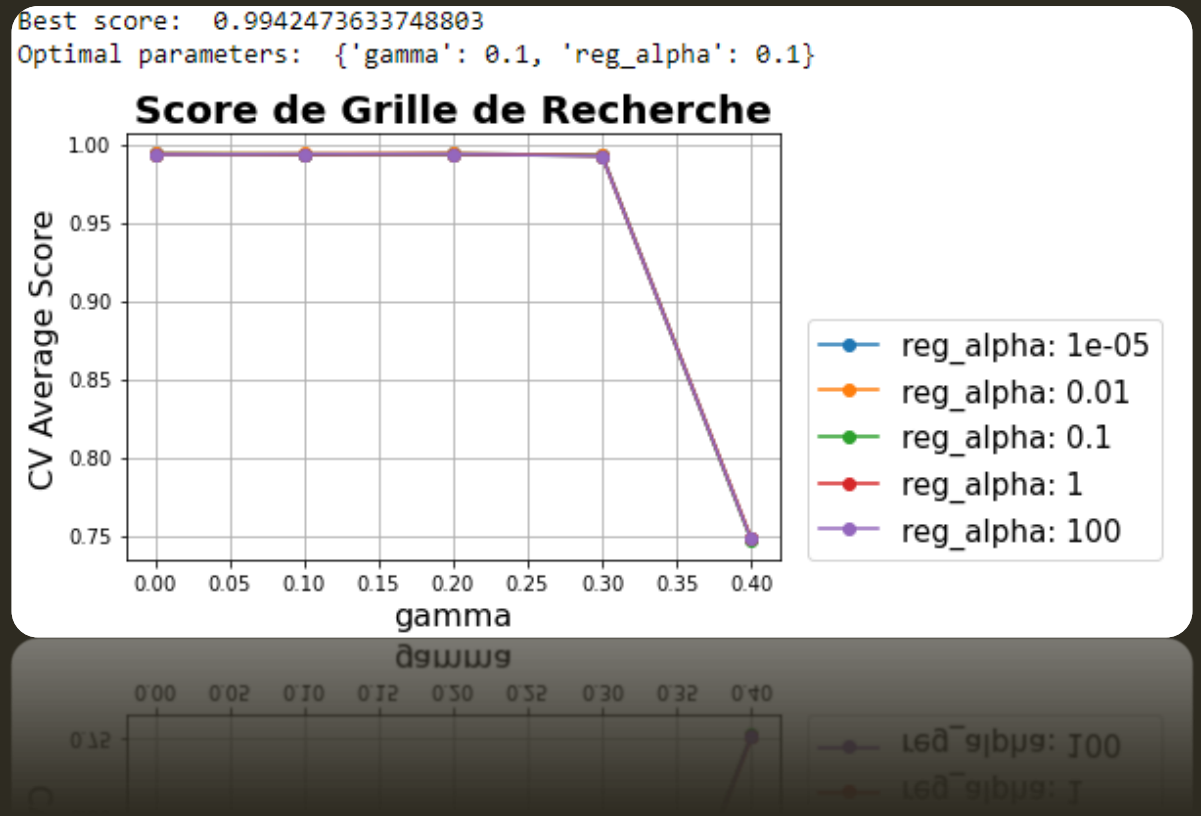
Pour XGBC :

```
Best score:  0.9938638542665389
Optimal parameters:  {'max_depth': 9, 'min_child_weight': 1}
```



3. PRÉDICTION ET MODÈLES

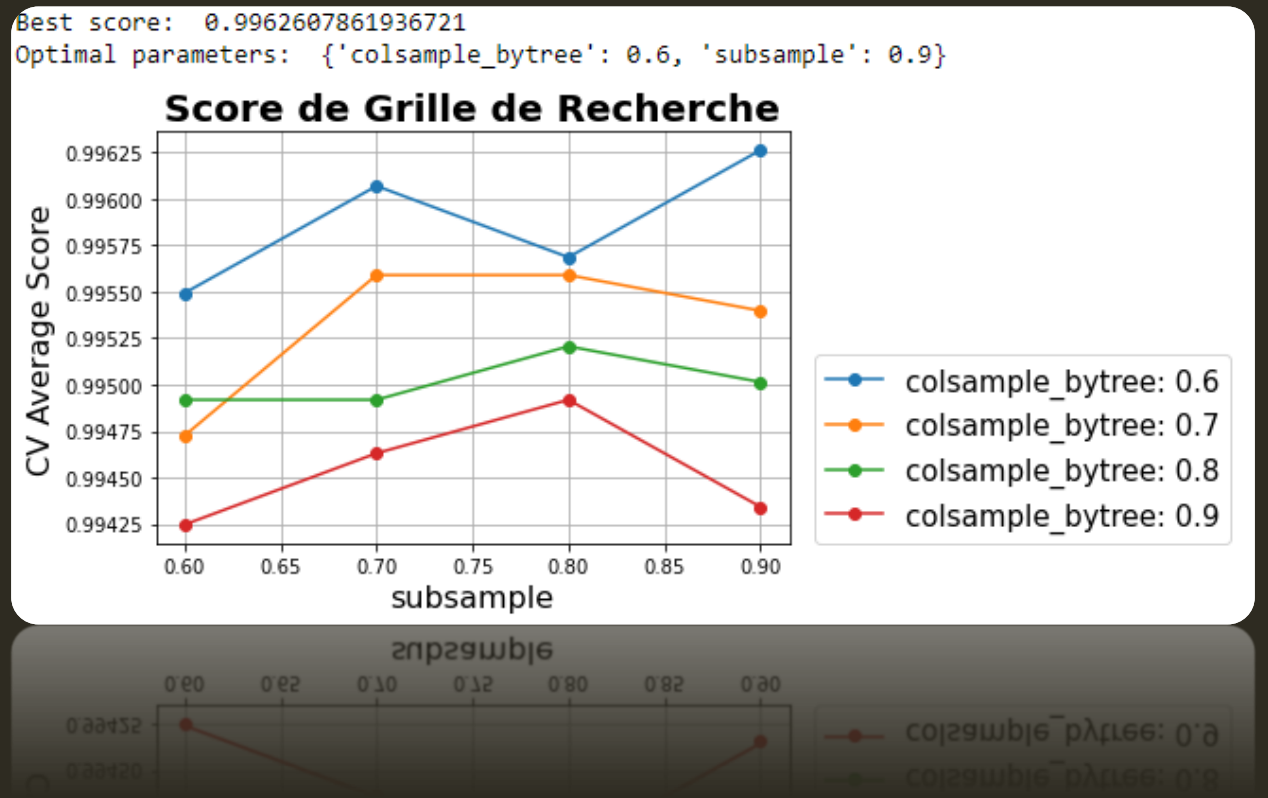
Pour XGBC (2) :



3. PRÉDICTION ET MODÈLES

Pour XGBC (3) :

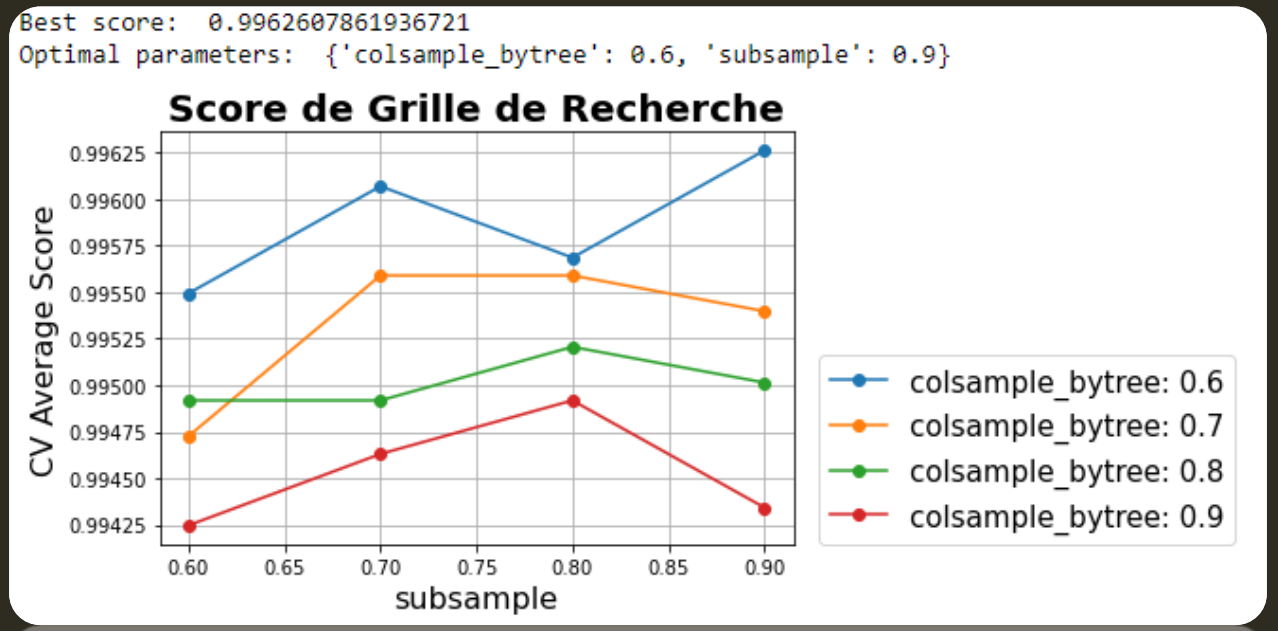
```
Best score:  0.9962607861936721
Optimal parameters:  {'colsample_bytree': 0.6, 'subsample': 0.9}
```



3. PRÉDICTION ET MODÈLES

Pour XGBC (4) :

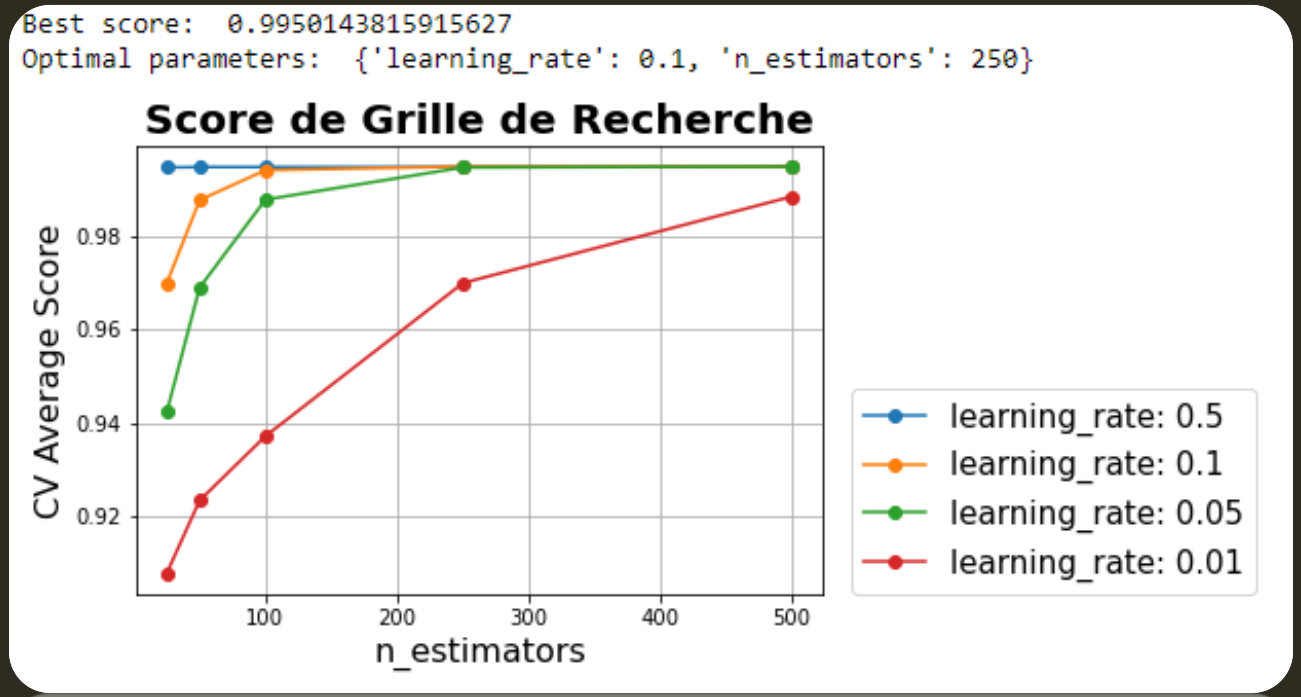
```
Best score:  0.9962607861936721
Optimal parameters:  {'colsample_bytree': 0.6, 'subsample': 0.9}
```



3. PRÉDICTION ET MODÈLES

Pour XGBC (5) :

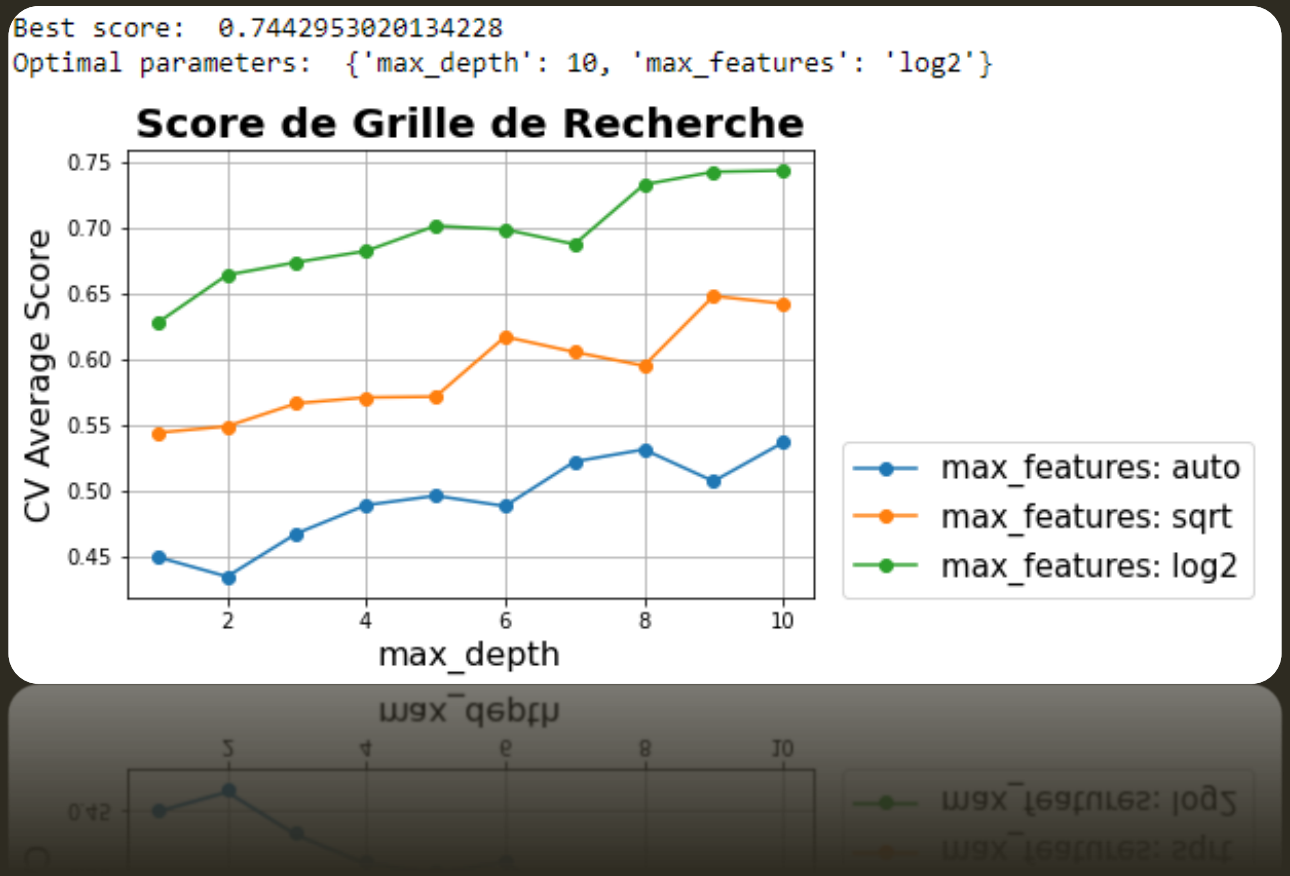
```
Best score:  0.9950143815915627
Optimal parameters:  {'learning_rate': 0.1, 'n_estimators': 250}
```



3. PRÉDICTION ET MODÈLES

Pour DTC :

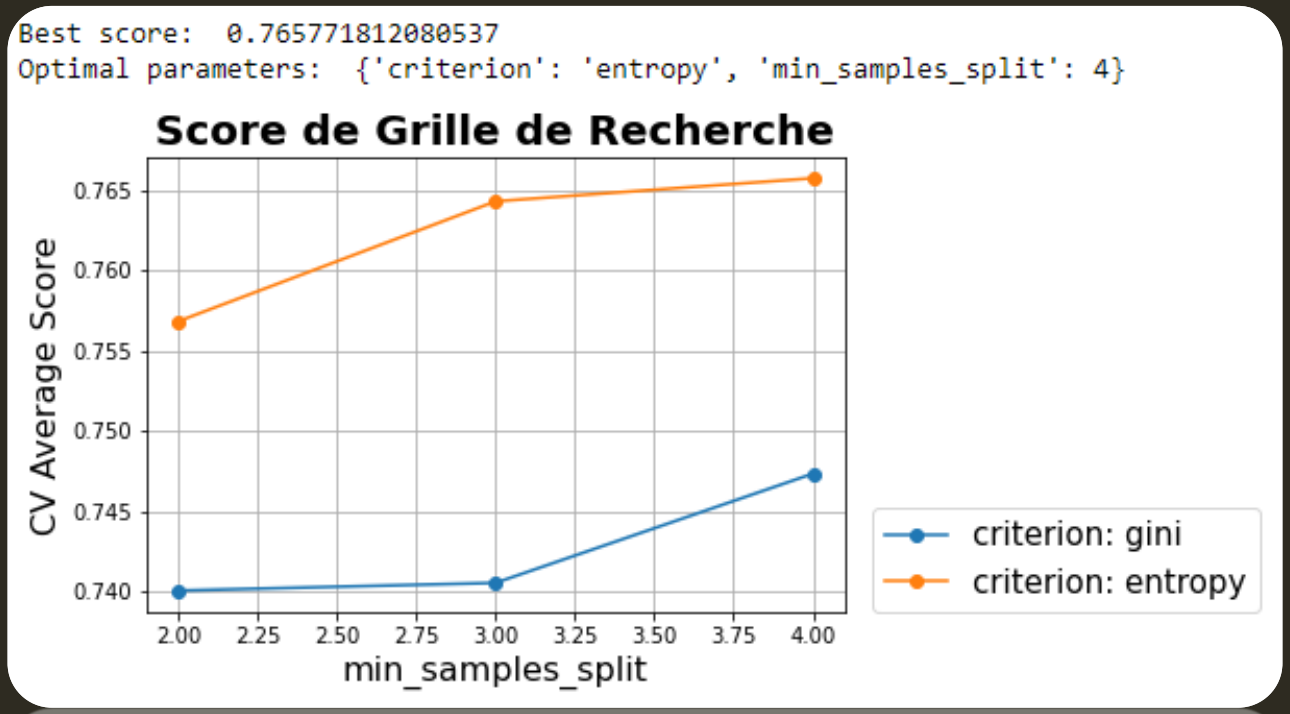
```
Best score:  0.7442953020134228
Optimal parameters:  {'max_depth': 10, 'max_features': 'log2'}
```



3. PRÉDICTION ET MODÈLES

Pour DTC (2) :

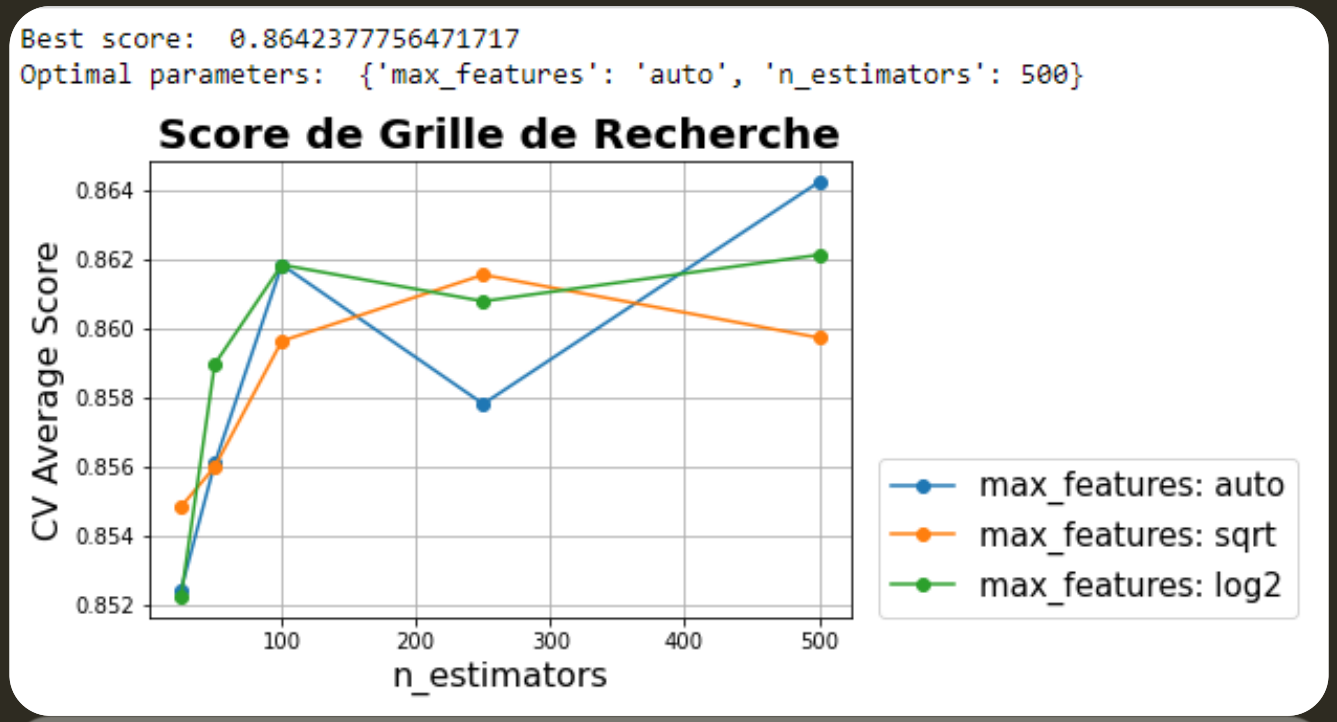
```
Best score:  0.765771812080537
Optimal parameters:  {'criterion': 'entropy', 'min_samples_split': 4}
```



3. PRÉDICTION ET MODÈLES

Pour RFC :

```
Best score:  0.8642377756471717
Optimal parameters:  {'max_features': 'auto', 'n_estimators': 500}
```



3. PRÉDICTION ET MODÈLES

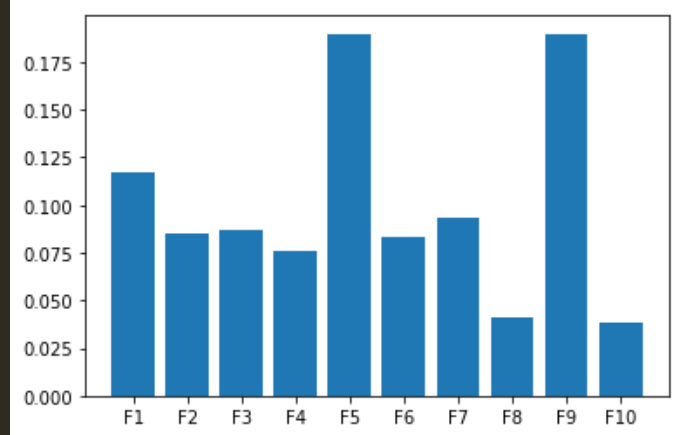
Nous avons ensuite évalué les résultats de chaque modèle.

Pour XGBC :

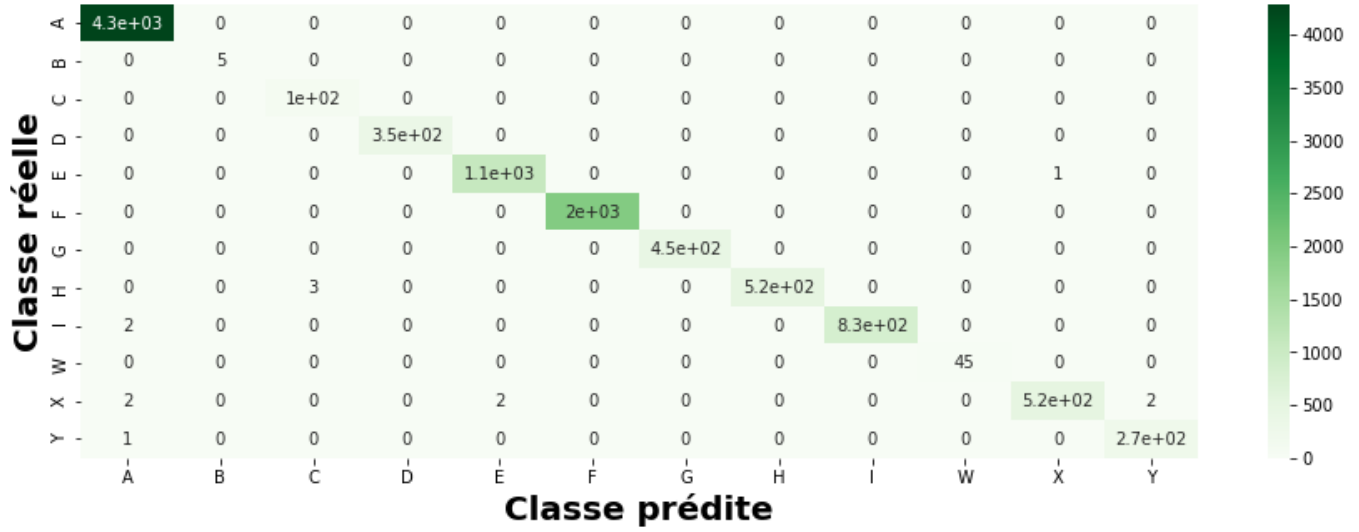
Accuracy on training set:
1.0
Accuracy on testing set:
0.9987544313500047
Classification Report:

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	4286
1.0	1.00	1.00	1.00	5
2.0	0.97	1.00	0.99	103
3.0	1.00	1.00	1.00	353
4.0	1.00	1.00	1.00	1095
5.0	1.00	1.00	1.00	1962
6.0	1.00	1.00	1.00	447
7.0	1.00	0.99	1.00	520
8.0	1.00	1.00	1.00	832
9.0	1.00	1.00	1.00	45
10.0	1.00	0.99	0.99	522
11.0	0.99	1.00	0.99	267
accuracy			1.00	10437
macro avg	1.00	1.00	1.00	10437
weighted avg	1.00	1.00	1.00	10437

Importances des features:



Confusion Matrix:



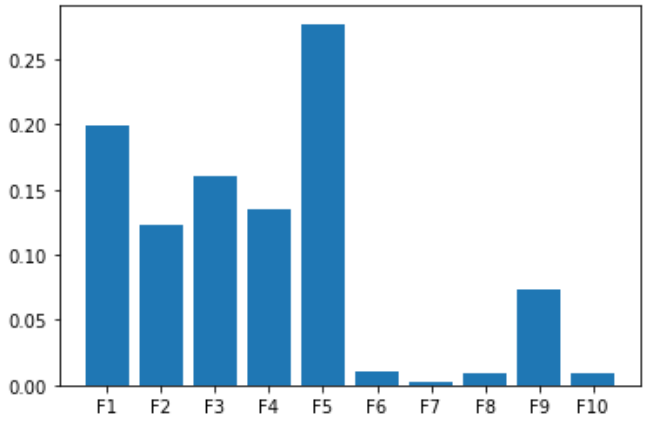
3. PRÉDICTION ET MODÈLES

Pour DTC :

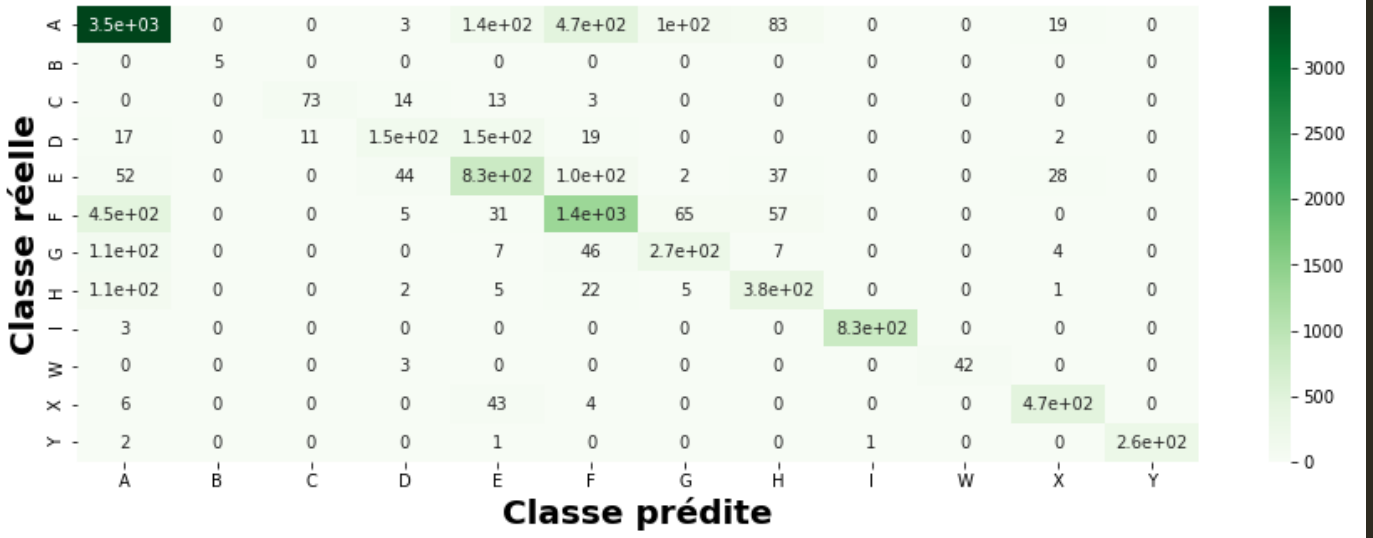
Accuracy on training set:
0.7969319271332694
Accuracy on testing set:
0.7793427230046949
Classification Report:

	precision	recall	f1-score	support
0.0	0.82	0.81	0.82	4286
1.0	1.00	1.00	1.00	5
2.0	0.87	0.71	0.78	103
3.0	0.68	0.43	0.53	353
4.0	0.68	0.76	0.72	1095
5.0	0.67	0.69	0.68	1962
6.0	0.61	0.60	0.61	447
7.0	0.67	0.73	0.70	520
8.0	1.00	1.00	1.00	832
9.0	1.00	0.93	0.97	45
10.0	0.90	0.90	0.90	522
11.0	1.00	0.99	0.99	267
accuracy			0.78	10437
macro avg	0.83	0.80	0.81	10437
weighted avg	0.78	0.78	0.78	10437

Importances des features:



Confusion Matrix:



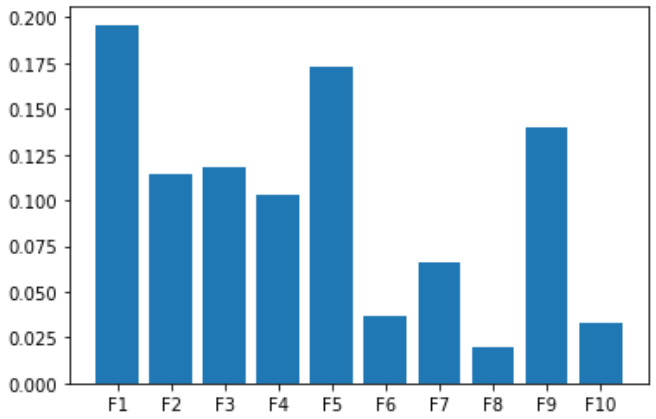
3. PRÉDICTION ET MODÈLES

Pour RFC :

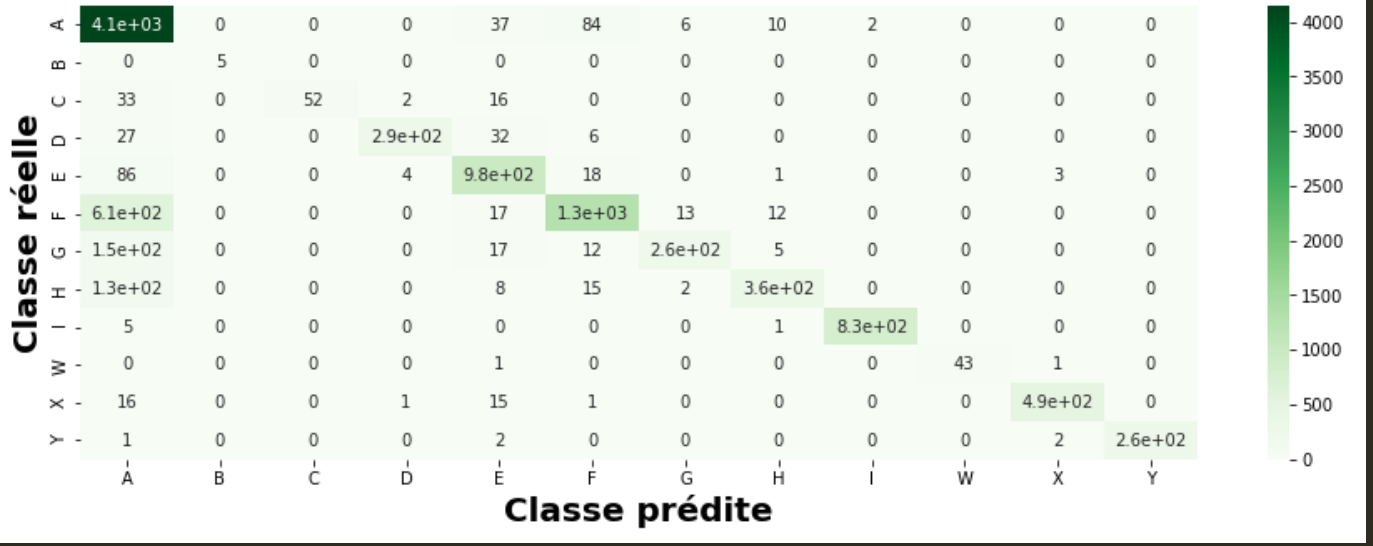
Accuracy on training set:
0.8939597315436242
Accuracy on testing set:
0.8649995209351347
Classification Report:

	precision	recall	f1-score	support
0.0	0.80	0.97	0.87	4286
1.0	1.00	1.00	1.00	5
2.0	1.00	0.50	0.67	103
3.0	0.98	0.82	0.89	353
4.0	0.87	0.90	0.88	1095
5.0	0.91	0.67	0.77	1962
6.0	0.93	0.58	0.71	447
7.0	0.93	0.70	0.80	520
8.0	1.00	0.99	1.00	832
9.0	1.00	0.96	0.98	45
10.0	0.99	0.94	0.96	522
11.0	1.00	0.98	0.99	267
accuracy			0.86	10437
macro avg	0.95	0.83	0.88	10437
weighted avg	0.88	0.86	0.86	10437

Importances des features:



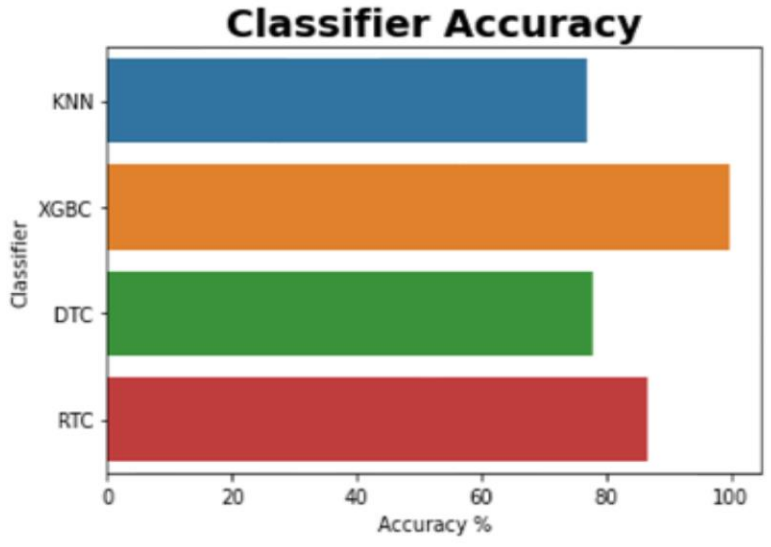
Confusion Matrix:

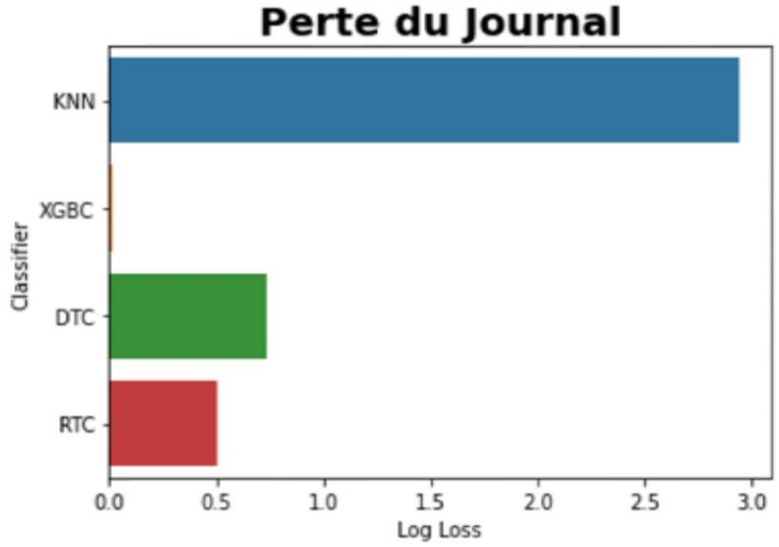


3. PRÉDICTION ET MODÈLES

Enfin, nous avons comparé ces différents modèles de manière simplifiée :

```
==============================
KNN
****Results****
Accuracy: 77.0432%
Log Loss: 2.9438334170667506
==============================
XGBC
****Results****
Accuracy: 99.8754%
Log Loss: 0.01464634974050203
==============================
DTC
****Results****
Accuracy: 77.9343%
Log Loss: 0.7330596388436019
==============================
RTC
****Results****
Accuracy: 86.5000%
Log Loss: 0.5061497729471257
==============================
```



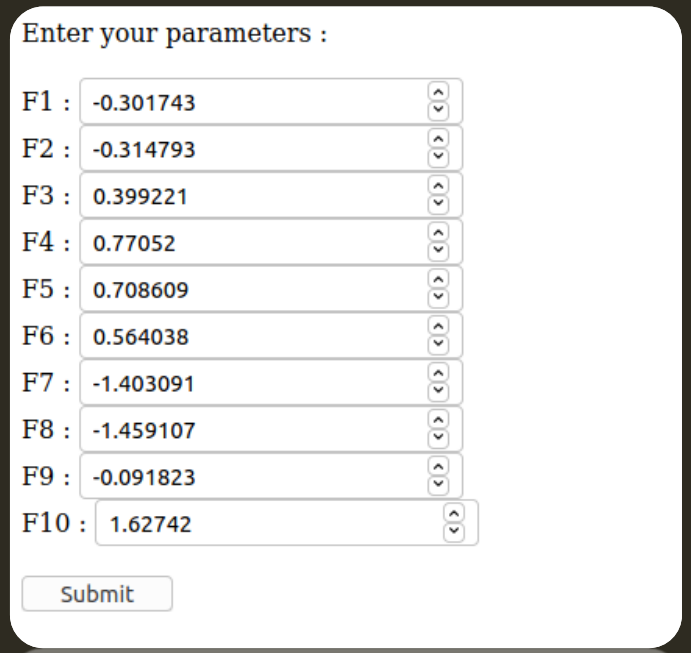


Nous pouvons ainsi clairement déterminer que le modèle XGBC est le plus optimale pour notre dataset. C'est donc celui que nous utiliserons pour notre API Django.

4. API DJANGO

Après avoir exporté notre modèle avec pickle et créé notre API avec Django, nous avons réalisé une simple interface graphique pour montrer son fonctionnement :

L'utilisateur est tout d'abord invité à rentrer manuellement les différentes caractéristiques :



La réponse de la prédiction lui est ensuite envoyée au format JSON. Ici, nous pouvons voir que le copiste prédit est le n°11, ce qui correspond à la lettre X.