Université Abdelmalek Essaadi
Faculté des sciences et technique -Tanger-
Département Génie Informatique

Demi-Module : Machine Learning

Rapport : Projet Machine Learning

Kaggle Competition

Football Match Probability Prediction with LogisticRegression

Réalisé par :

Younes ET-TOUIL

Abdelmonim DAKKON

Encadré par :

Pr. Lotfi ELAACHAK

Table des matières

Introduction : 3

I. Kaggle Compeitition : Football Match Probability Prediction. 4

1. Kaggle community: 4

2. Fonctionnement des compétitions Kaggle : 4

3. Compétitions Kaggle : Football Match Probability Prediction 4

3.1 Description du Compétitions 4

3.2 Description de l'ensemble de données : 5

II. Les algorithmes et les techniques utilisés : 6

1. L'algorithme Régression logistique : 6

1.1 Le mode de fonctionnement : 7

2. Sélection des caractéristiques "Features Selection" : 7

2.1 Sélection univariée « Univariate Selection, US » 8

2.2 Analyse des composants principaux "Principal Component Analysis" 8

2.3 Importance des caractéristiques "Feature Importance, FI" 8

III. NoteBook de Compétition 'Football Match Probability Prediction' : 9

1. Importer des modules et charger les données 9

2. Prétraitement des données : 9

3. Sélection des caractéristiques "Features Selection" : 12

4. Analyse en composantes principales (PCA) : 14

5. Feature Importance, FI : 16

6. Entraînement à l'aide de la régression logistique : 16

Conclusion : 18

Introduction :

Le machine learning (ML), ou apprentissage machine, fait partie des principales technologies d'intelligence artificielle. Il permet de réaliser des prédictions en se basant sur un modèle entraîné à partir d'un historique de données qui pourra évoluer dans le temps. Là où un programme traditionnel exécute des instructions, un algorithme de machine learning améliore ses performances au fur et à mesure de son apprentissage. Plus on le "nourrit" de données, plus il devient précis.

Pour décrire son modèle d'apprentissage, le machine learning emploie des algorithmes statistiques ou encore des réseaux de neurones. Dans les années 2010, le machine learning a atteint un Momentum avec l'avènement du big data et la progression des capacités de calcul (et notamment la montée en puissance des GPU). Le big data est en effet indispensable pour entraîner des modèles sur les vastes volumes de données nécessaires au traitement automatique du langage ou à la reconnaissance d'images.

On distingue **les algorithmes de machine learning supervisés** des **algorithmes de machine learning non supervisés**. Côté apprentissage supervisé, les données d'entrainement sont préalablement annotées ou étiquetée. Objectif : recourir à une base d'apprentissage représentative qui permette d'aboutir à un modèle capable de généraliser, c'est-à-dire de réaliser ensuite des prédictions correctes sur des données non présentes dans la base d'apprentissage de départ. Dans le domaine de l'apprentissage supervisé, on retrouve les algorithmes de classification, de régression linéaire, **de régression logistique**, les arbres de décision, ou encore les forêts aléatoires.

Quant à l'apprentissage non supervisé, il décode les informations de contexte des données d'entrainement et la logique qui en découle, sans recourir à une source de connaissances préétablie. Les données ne sont ni annotées ni étiquetées. Dans cette catégorie, on retrouve les algorithmes de clustering (comme K-means) conçus pour partager les données en groupes similaires. Ils peuvent permettre par de réaliser des regroupements par typologies de clients, en fonction de caractéristiques de profils, de comportements d'achat similaires...

I. Kaggle Compeitition : Football Match Probability Prediction.

1. Kaggle community:

Kaggle est une plateforme web organisant des compétitions en science des données appartenant à Google. Sur cette plateforme, les entreprises proposent des problèmes en science des données et offrent un prix aux datalogistes obtenant les meilleures performances. L'entreprise a été fondée en 2010 par Anthony Goldbloom.

2. Fonctionnement des compétitions Kaggle :

- L'animateur de la compétition prépare les données ainsi qu'une description du problème. Kaggle propose un service de conseil, encadre le concours et anonymise les données…
- Les participants expérimentent avec différentes techniques et s'affrontent pour produire le meilleur modèle. Dans la plupart des compétitions, les observations sont notées immédiatement (la note se base sur leur valeur prédictive par rapport à un fichier de solution cachée) qui donne un classement en direct.
- À la fin de la compétition, l'animateur distribue le lot (argent, proposition de poste) en échange de "a worldwide, perpetual, irrevocable and royalty free license [...] to use the winning Entry", c'est-à-dire le droit d'utiliser gratuitement et sans limite de temps l'algorithme et le logiciel élaborés, de façon "non exclusive sauf indication contraire".

Aux côtés de ses concours publics, l'entreprise propose également des concours privés limités aux meilleurs participants de Kaggle et pour des groupes universitaires.

3. Compétitions Kaggle : Football Match Probability Prediction

Octosport et Sportmonks sont fiers de présenter leur premier concours de pronostics footballistiques. Vous savez coder en python et êtes passionné de football ? Vous souhaitez tester vos connaissances en machine learning, séquence learning ou réseaux de neurones ? Vous voulez battre les performances du bookmaker ? Vous êtes au bon endroit.

3.1 Description du Compétitions

Le football est au cœur de la science des données depuis plus d'une décennie. Si les algorithmes d'aujourd'hui se concentrent sur la détection d'événements, le style de joueur ou l'analyse d'équipe, prédire les résultats d'un match reste un défi ouvert.

Prédire les résultats d'un match entre deux équipes dépend principalement (mais pas seulement) de leur forme actuelle. La forme d'une équipe peut être considérée comme sa séquence récente de résultats par rapport aux autres équipes. Ainsi, les probabilités de match entre deux équipes peuvent être différents comptes tenus de leur calendrier.

Cette compétition consiste à prédire les probabilités de plus de 150 000 résultats de match en utilisant la séquence récente de 10 matchs des équipes.

3.2 Description de l'ensemble de données :

L'ensemble de données contient plus de 150000 matchs de football mondiaux historiques de 2019 à 2021, avec plus de 860 ligues et 9500 équipes. Votre objectif est de prédire les probabilités pour chaque résultat de match dans les colonnes cibles : les probabilités de gagner à domicile, de faire match nul et de gagner à l'extérieur.

Signification Features, Home et Away

Les caractéristiques fournies sont divisées en deux parties : les caractéristiques descriptives et les caractéristiques historiques. Les caractéristiques descriptives sont des descriptions ponctuelles de la correspondance qui doit être prédite. Les caractéristiques historiques contiennent des informations passées sur les 10 matchs précédemment joués par l'équipe à domicile et l'équipe à l'extérieur.

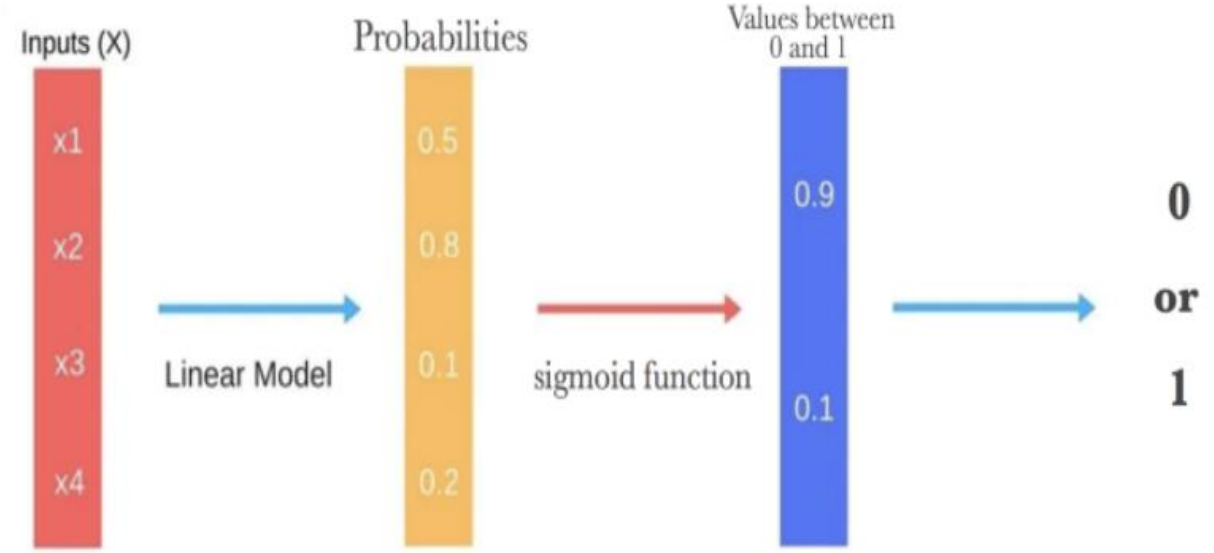
Home et Away sont utilisés par convention pour décrire un match entre deux équipes Home - Away. Par exemple Manchester City - Chelsea. Dans ce cas, Manchester City joue à domicile. Pour les coupes, l'équipe locale ne joue pas nécessairement à domicile, mais l'information n'est pas disponible dans l'ensemble de données.

Notez que pour les données historiques, pour savoir quelle équipe joue à domicile, veuillez-vous référer à la colonne de données historiques is_play_home car l'équipe à domicile et l'équipe à l'extérieur peuvent avoir joué à domicile ou à l'extérieur lors de leurs 10 derniers matchs.

II. Les algorithmes et les techniques utilisés :

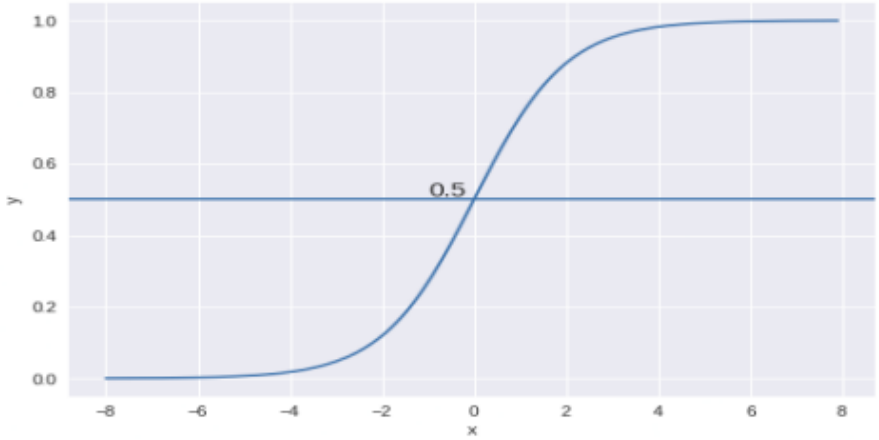
1. L'algorithme Régression logistique :

La régression logistique est un algorithme de classification utilisé pour attribuer des observations à un ensemble discret de classes. Certains des exemples de problèmes de classification sont les spams par courrier électronique ou non, les transactions en ligne frauduleuses ou non, les tumeurs malignes ou bénignes.



On peut dire qu'une régression logistique est un modèle de régression linéaire, mais il utilise une fonction de coût plus complexe. Cette fonction de coût peut être définie comme étant la «fonction sigmoïde» ou encore appelée «fonction logistique» au lieu d'une fonction linéaire.

$$\gamma(t) = \frac{1}{1 + e^{-t}}$$



En régression linéaire, la sortie est la somme pondérée des entrées. La régression logistique est une régression linéaire généralisée en ce sens que nous ne produisons pas directement la somme pondérée des entrées, mais que nous la transmettons via une fonction qui peut mapper toute valeur réelle comprise entre 0 et 1.

1.1 Le mode de fonctionnement :

La régression logistique mesure la relation entre la variable dépendante (notre class, ce que nous voulons prédire) et la ou les variables indépendantes (nos caractéristiques), en estimant les probabilités à l'aide de sa fonction logistique sous-jacente.

Ces probabilités doivent ensuite être transformées en valeurs binaires pour pouvoir réellement prédire. C'est la tâche de la fonction logistique, également appelée fonction sigmoïde. La fonction Sigmoïde est capable de prendre n'importe quel nombre réel et la mapper en une valeur comprise entre 0 et 1, mais jamais exactement à ces limites. Ces valeurs comprises entre 0 et 1 seront ensuite transformées en 0 ou 1 à l'aide d'un classifieur de seuil.

Le processus de fonctionnement de régression logistique va suivre ces étapes :

- Calculer la fonction logistique.
- Calculer les coefficients d'un modèle de régression logistique utilisant une descente de gradient stochastique.
- Faire des prédictions en utilisant un modèle de régression logistique.

2. Sélection des caractéristiques "Features Selection" :

Les caractéristiques de données qu'on utilise pour former des modèles d'apprentissage machine ont une énorme influence sur les performances. c'est-à-dire ses caractéristiques non pertinentes ou partiellement pertinentes peuvent avoir un impact négatif sur les performances des modèles déjà établis.

La sélection des caractéristiques est un processus dans lequel on sélectionne automatiquement les fonctionnalités des données qui contribuent le plus à la variable de prédiction ou à la sortie. La présence des caractéristiques non pertinentes dans des données peut réduire la précision de nombreux modèles, en particulier les algorithmes linéaires tels que la régression linéaire et logistique.

La sélection des caractéristiques avant la modélisation des données présente trois avantages :

- Réduit les Overfitting : Moins de données redondantes signifie moins d'opportunités pour prendre des décisions basées sur le bruit.

- Améliore la précision : moins de données trompeuses signifie que la précision de la modélisation est améliorée.

- Réduit le temps de l'entrainement : moins de données signifie que les algorithmes s'entraînent plus rapidement.

Pour le langage Python la bibliothèque la plus réputée pour l 'apprentissage machine est sklearn, cette bibliothèque plusieurs méthode pour la sélection les caractéristiques de données, Dans cette section on va répertorier 4 recettes de sélection de fonctionnalités pour l'apprentissage automatique en Python « Sélection univariée, Élimination récursive de caractéristiques, Analyse des composants principaux et Importance des caractéristiques ».

2.1 Sélection univariée « Univariate Selection, US »

Les tests statistiques peuvent être utilisés pour sélectionner les entités qui ont la relation la plus forte avec la variable de sortie. La bibliothèque scikit-learn fournit la classe SelectKBest qui peut être utilisée avec une suite de différents tests statistiques pour sélectionner un nombre spécifique de fonctionnalités.

(Voire le code python dans partie de code de la solution)

2.2 Analyse des composants principaux "Principal Component Analysis"

L'analyse en composantes principales (ou ACP) utilise l'algèbre linéaire pour transformer l'ensemble de données en une forme compressée. Généralement, cela s'appelle une technique de réduction des données. Une propriété de PCA est que vous pouvez choisir le nombre de dimensions ou de composant principal dans le résultat transformé.

(Voire le code python dans partie de code de la solution)

2.3 Importance des caractéristiques "Feature Importance, FI"

Des arbres de décision comme Random Forest et Extra Trees peuvent être utilisés pour estimer l'importance des entités

III. NoteBook de Compétition 'Football Match Probability Prediction' :

1. Importer des modules et charger les données

Commençons par importer les modules, la formation et l'ensemble de test. L'ensemble de test contient les mêmes colonnes que l'ensemble d'apprentissage sans la cible.

```
# data processing
import pandas as pd
# linear algebra
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

```
# load the data
train = pd.read_csv("../input/football-match-probability-prediction/train.csv")
test = pd.read_csv("../input/football-match-probability-prediction/test.csv")
```

```
# show the first 5 rows
train.head()
```

	id	target	is_cup	home_team_history_is_play_home_1	home_team_history_is_play_home_2	home_team_history_i
0	11906497	0	0	0.0	1.0	
1	11984383	2	0	1.0	0.0	
2	11983301	1	0	0.0	1.0	
3	11983471	0	0	0.0	0.0	
4	11883005	2	0	0.0	1.0	

5 rows × 143 columns

2. Prétraitement des données :

Le prétraitement des données peut faire référence à la manipulation ou à la suppression de données avant qu'elles ne soient utilisées afin d'assurer ou d'améliorer les performances.

La fonction qui renvoie les colonnes contient la chaîne où nous avons écrit :

```
#function that return columns contient the string where we written
def features_filtred(data,column_name):
    features_filtred = [_ for _ in data.columns if str(column_name) in _]
    return features_filtred
```

Nous supprimons toutes les colonnes contient _id ,_name,_date

```
# we drop all columns contient _id ,_name,_date
#from train-datasets
all_id_features = features_filtred(train,"_id")
train = train.drop(all_id_features, 1)

all_name_features1 = features_filtred(train,"_name")
train = train.drop(all_name_features1, 1)

all_date_features2 = features_filtred(train,"date")
train = train.drop(all_date_features2, 1)

#from test-datasets
all_id_features3 = features_filtred(test,"_id")
test = test.drop(all_id_features3, 1)

all_name_features4 = features_filtred(test,"_name")
test = test.drop(all_name_features4, 1)
all_date_features5 = features_filtred(test,"date")
test = test.drop(all_date_features5, 1)
```

Supprimer les valeurs manquantes des colonnes et les remplacer par ses moyennes

```
#sum of missing values in every column in our datasets
train.isnull().sum()
```

```
id                                    0
target                                0
is_cup                                1
home_team_history_is_play_home_1   1159
home_team_history_is_play_home_2   2451
                                   ...
away_team_history_coach_6         23525
away_team_history_coach_7         24377
away_team_history_coach_8         25219
away_team_history_coach_9         26150
away_team_history_coach_10        27129
Length: 143, dtype: int64
```

```
#Replace the missing values in each column with its mean
train["is_cup"].fillna(True, inplace=True)
train.fillna(train.mean(), inplace=True)
test.fillna(test.mean(), inplace=True)
```

```
train.isnull().sum()
```

```
id                                0
target                            0
is_cup                            0
home_team_history_is_play_home_1  0
home_team_history_is_play_home_2  0
                                 ..
away_team_history_coach_6         0
away_team_history_coach_7         0
away_team_history_coach_8         0
away_team_history_coach_9         0
away_team_history_coach_10        0
Length: 143, dtype: int64
```

Encodage des deux caractéristiques ['target'] et ['is_cup']

```
#Encoding the two features ['target'] and ['is_cup']
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()

test['is_cup']  = labelencoder.fit_transform(test['is_cup'])
train['is_cup'] = labelencoder.fit_transform(train['is_cup'])
train['target'] = labelencoder.fit_transform(train['target'])
```

```
train['is_cup'].unique()
test['is_cup'].unique()
```

```
array([0, 1])
```

```
train['target'].unique()
```

```
array([0, 2, 1])
```

Séparation des données des trains à *Input* et *Output*

```
#split train data to Input "X" and Output "Y"
ID = train['id']
Y = train['target']
X = train.iloc[:, 2:]
```

```
X.head()
```

	is_cup	home_team_history_is_play_home_1	home_team_history_is_play_home_2	home_team_history_is_play_home_3	ho
0	0	0.0	1.0	0.0	
1	0	1.0	0.0	1.0	
2	0	0.0	1.0	0.0	
3	0	0.0	0.0	1.0	
4	0	0.0	1.0	0.0	

5 rows × 141 columns

3. Sélection des caractéristiques "Features Selection" :

La bibliothèque scikit-learn fournit la classe SelectKBest qui peut être utilisée avec une suite de différents tests statistiques pour sélectionner un nombre spécifique de fonctionnalités.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
SKB = SelectKBest(score_func=chi2, k=4)
fit = SKB.fit(X, Y)
np.set_printoptions(precision=3)
print(fit.scores_)
features = fit.transform(X)
print(features[0:5,:])
X.columns
```

```
[9.151e+01 7.244e+00 2.526e+00 3.250e+00 1.992e-01 1.756e+00 3.820e+00
 8.782e-03 4.019e+00 2.759e-01 4.881e-01 5.217e+01 4.293e+01 1.939e+01
 3.586e+01 2.884e+01 5.658e+01 4.985e+01 3.497e+01 5.600e+01 4.095e+01
 6.997e+02 6.633e+02 7.087e+02 5.057e+02 5.625e+02 5.171e+02 3.997e+02
 3.376e+02 3.472e+02 2.813e+02 9.979e+02 8.785e+02 8.396e+02 6.492e+02
 4.686e+02 6.186e+02 4.235e+02 4.058e+02 3.685e+02 3.558e+02 1.659e+03
 1.621e+03 1.472e+03 1.260e+03 1.314e+03 1.183e+03 1.068e+03 9.148e+02
 8.415e+02 7.893e+02 1.967e+03 1.734e+03 1.539e+03 1.358e+03 1.170e+03
 1.127e+03 9.740e+02 8.985e+02 8.523e+02 7.000e+02 1.590e+09 1.504e+09
 2.615e+09 2.295e+09 2.147e+09 1.836e+09 1.765e+09 1.666e+09 1.691e+09
 1.331e+09]
[[  468200.   468200.   468200.   468200.]
 [22169161. 22169161. 22169161. 22169161.]
 [  456313.   456313.   456313.   456313.]
 [  465797.   465797.   465797.   465797.]
 [  426870.   426870.   426870.   426870.]]
Index(['is_cup', 'home_team_history_is_play_home_1',
       'home_team_history_is_play_home_2', 'home_team_history_is_play_home_3',
       'home_team_history_is_play_home_4', 'home_team_history_is_play_home_5',
       'home_team_history_is_play_home_6', 'home_team_history_is_play_home_7',
       'home_team_history_is_play_home_8', 'home_team_history_is_play_home_9',
       ...
       'away_team_history_coach_1', 'away_team_history_coach_2',
       'away_team_history_coach_3', 'away_team_history_coach_4',
       'away_team_history_coach_5', 'away_team_history_coach_6',
       'away_team_history_coach_7', 'away_team_history_coach_8',
       'away_team_history_coach_9', 'away_team_history_coach_10'],
      dtype='object', length=141)
```

Vous pouvez voir les scores pour chaque attribut et les attributs choisis (ceux avec les scores les plus élevés). Plus haut)

```
# we drop the features with the low scores
all_history_coach_features = features_filtred(X,"history_coach_")
X = X.drop(all_history_coach_features, 1)

all_history_rating_features = features_filtred(X,"rating_")
X = X.drop(all_history_rating_features, 1)

all_history_coach_features_test = features_filtred(test,"history_coach_")
test = test.drop(all_history_coach_features_test, 1)

all_history_rating_features_test = features_filtred(test,"rating_")
test = test.drop(all_history_rating_features_test, 1)
```

Créer des sous-tableaux pour les 'Features'de Home_Team / Away_Team

```
#create sub-tables for features of Home team
home_team_history_is_play_home = features_filtred(train,"home_team_history_is_play_h
X_HTPH = X[home_team_history_is_play_home]

home_team_history_is_cup = features_filtred(train,"home_team_history_is_cup_")
X_HTC = X[home_team_history_is_cup]

home_team_history_goal = features_filtred(train,"home_team_history_goal_")
X_HTG = X[home_team_history_goal]

home_team_history_opponent_goal = features_filtred(train,"home_team_history_opponent
X_HTOG = X[home_team_history_opponent_goal]


#create sub-tables for features of Away team

away_team_history_is_play_home = features_filtred(train,"away_team_history_is_play_h
X_ATPH = X[away_team_history_is_play_home]

away_team_history_is_cup = features_filtred(train,"away_team_history_is_cup_")
X_ATC = X[away_team_history_is_cup]

away_team_history_goal = features_filtred(train,"away_team_history_goal_")
X_ATG = X[away_team_history_goal]

away_team_history_opponent_goal = features_filtred(train,"away_team_history_opponent
X_ATOG = X[away_team_history_opponent_goal]
```

```
X_HTPH_test
```

	home_team_history_is_play_home_1	home_team_history_is_play_home_2	home_team_history_is_play_home_3	home_
0	0.0	0.0	1.0	
1	0.0	1.0	1.0	
2	1.0	0.0	1.0	
3	0.0	0.0	1.0	
4	0.0	0.0	1.0	
...	...	...	...	
72706	1.0	0.0	1.0	
72707	0.0	1.0	0.0	
72708	1.0	1.0	0.0	
72709	0.0	0.0	0.0	
72710	0.0	1.0	0.0	

72711 rows × 10 columns

On voit que on obtient 10 colonnes, c-à-d chaque sub-table a 10 colonnes et 72711 lignes

4. Analyse en composantes principales (PCA) :

Une propriété de PCA est que vous pouvez choisir le nombre de dimensions ou le composant principal dans le résultat transformé. Où nous réduisons les 10 fonctionnalités qu'on a obtient dans la cellule précédant à une seule fonctionnalité.

```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=1)

#PCA for Home_Team_Features
X_HTPH_C = pca.fit_transform(X_HTPH)
X_HTC_C = pca.fit_transform(X_HTC)
X_HTG_C = pca.fit_transform(X_HTG)
X_HTOG_C = pca.fit_transform(X_HTOG)

#PCA for Home_Team_Features
X_ATPH_C = pca.fit_transform(X_ATPH)
X_ATC_C = pca.fit_transform(X_ATC)
X_ATG_C = pca.fit_transform(X_ATG)
X_ATOG_C = pca.fit_transform(X_ATOG)
```

La même chose on fait avec sub-tables de test.

Créer une nouvelle fonctionnalité qui remplace les 10 anciennes fonctionnalités via PCA

```
# create a new feature that replaces the 10 old features through PCA
X["home_team_is_play_home"] = X_HTPH_C
home_team_history_is_play_home = features_filtred(train,"home_team_history_is_play_h
X = X.drop(home_team_history_is_play_home,1)

X["home_team_is_cup"] = X_HTC_C
home_team_history_is_cup = features_filtred(train,"home_team_history_is_cup_")
X = X.drop(home_team_history_is_cup,1)

X["home_team_goal"] = X_HTG_C
home_team_history_goal = features_filtred(train,"home_team_history_goal_")
X = X.drop(home_team_history_goal,1)

X["home_team_opponent_goal"] = X_HTOG_C
home_team_history_opponent_goal = features_filtred(train,"home_team_history_opponent
X = X.drop(home_team_history_opponent_goal,1)

X["away_team_is_play_home"] = X_ATPH_C
away_team_history_is_play_home = features_filtred(train,"away_team_history_is_play_h
X = X.drop(away_team_history_is_play_home,1)

X["away_team_is_cup"] = X_ATC_C
away_team_history_is_cup = features_filtred(train,"away_team_history_is_cup_")
X = X.drop(away_team_history_is_cup,1)

X["away_team_goal"] = X_ATG_C
away_team_history_goal = features_filtred(train,"away_team_history_goal_")
X = X.drop(away_team_history_goal,1)

X["away_team_opponent_goal"] = X_ATOG_C
away_team_history_opponent_goal = features_filtred(train,"away_team_history_opponent
X = X.drop(away_team_history_opponent_goal,1)
```

La même chose on fait avec sub-tables de test.

```
X.head(3)
```

	is_cup	home_team_is_play_home	home_team_is_cup	home_team_goal	home_team_opponent_goal	away_team_is_play_
0	0	-1.441105	-0.264352	-0.471041	-1.071028	-0.2
1	0	0.259370	-0.264352	-1.541345	-1.053435	0.7
2	0	-1.441105	-0.264352	1.596867	0.860644	0.2

Donc on réduire nombre des colonnes de notre ensemble de données de 190 à 9.

5. Feature Importance, FI :

```
from sklearn.ensemble import ExtraTreesClassifier

model = ExtraTreesClassifier()
model.fit(X, Y)
print(model.feature_importances_)
```

```
[0.004 0.13  0.078 0.144 0.146 0.13  0.078 0.145 0.145]
```

Voilà les Features important de notre nouvel ensemble de données.

6. Entraînement à l'aide de la régression logistique :

Préparation des données pour commencer l'entraînement :

```
ID_test = test['id']
ID_train = ID
x_train = X
y_train = Y
x_test = test.iloc[:,1:]
```

Fit le modèle et faire la soumission :

```
from sklearn.linear_model import LogisticRegression

#using LogisticRegression for prediction
lg = LogisticRegression()
lg.fit(x_train, y_train)
```

```
LogisticRegression()
```

```
y_prop_pred = lg.predict_proba(x_test)
y_prop_pred
```

```
array([[0.152, 0.265, 0.583],
       [0.348, 0.313, 0.339],
       [0.266, 0.291, 0.444],
       ...,
       [0.535, 0.279, 0.186],
       [0.228, 0.259, 0.513],
       [0.305, 0.263, 0.432]])
```

Y_prob_pred contient les probabilités de prédiction.

Cree le fichier de la soumission :

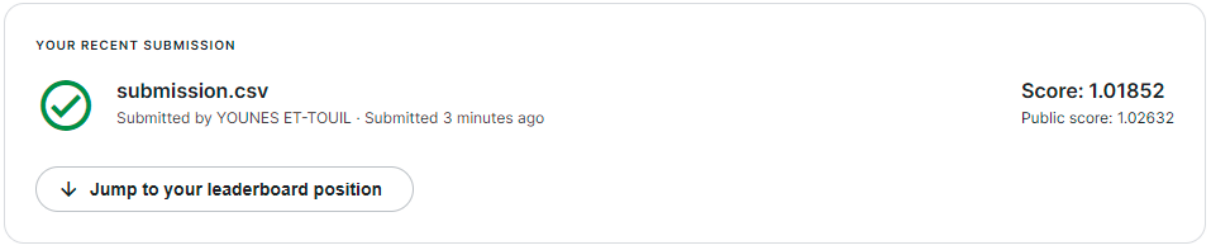
```
submission_test = pd.DataFrame(y_prop_pred,columns=['away', 'draw', 'home'])
submission = pd.DataFrame()
submission['id'] = test[['id']]
submission['away'] = submission_test['away']
submission['draw'] = submission_test['draw']
submission['home'] = submission_test['home']


submission[['id', 'home', 'away', 'draw']].to_csv('submission.csv', index=False)
```

```
submission
```

	id	away	draw	home
0	17761448	0.151924	0.264612	0.583464
1	17695487	0.347603	0.313016	0.339380
2	17715496	0.265670	0.290603	0.443727
3	17715493	0.433550	0.307243	0.259207
4	17715492	0.272592	0.322197	0.405210
...	...	...	...	...
72706	18450246	0.289695	0.303449	0.406856
72707	18164889	0.185666	0.263527	0.550808
72708	18449018	0.535029	0.278639	0.186331
72709	17958831	0.227972	0.259303	0.512725
72710	18441629	0.305022	0.263063	0.431915

72711 rows × 4 columns

Score après kaggle :



Conclusion :

Pour conclure, la réalisation de ce projet nous a permis de découvrir davantage la plateforme Kaggle en participant à une compétition, pour créer un modèle qui se rapporte à une problématique réelle.

Pour atteindre notre objectif, nous avons eu recours aux data processing, feature selection afin d'améliorer la précision de notre modèle, en plus de l'utilisation de différents algorithmes de machine learning, et de constater la déférence entre eux.

Lien répertoire GitHub :

https://github.com/younes-ettouil/Projet-Machine-Learning-kaggle-competition-

Lien NoteBook kaggle :

https://www.kaggle.com/code/younesettouil/football-match-probability-prediction-logistic-re