**Objectif du Projet**
L'objectif de ce projet est de créer un outil de prédiction utilisant Python pour déterminer quelles équipes se qualifieront aux différentes phases de la Ligue des Champions 2022/2023 (huitièmes de finale, quarts de finale, demi-finales, finale) et finalement, prédire le grand vainqueur.

**Étapes du Projet**

1) **Chargement et Préparation des Données :**

Je suis allé sur internet chercher un fichier open source sur le football. Je recherche un fichier avec des joueurs, des clubs ou des statistiques relative au football. Je suis tombé sur un forum internet partageant un lien vers Github. Sur ce site je trouve une multitude de fichier comportant les données de la ligue des champions depuis 2011.
https://openclassrooms.com/forum/sujet/base-de-donnees-football

Je choisis le fichier le plus récent : Les équipes qualifiées en LDC 2023/2023.
https://github.com/openfootball/europe-champions-league/blob/master/2022-23/cl.txt

Ensuite j'ai modifié le fichier TXT pour le convertir en fichier CSV puis je l'ai chargé sur Chat GPT car je ne maîtrise pas le codage Python.

> Pour un projet data à l'école je souhaite créer un outil de prédiction en utilisant python. Pour cela je dispose d'un fichier CSV avec l'ensemble des équipes qualifiées pour la ligue des champions de football 2022/2023. Je souhaite savoir quelle équipes vont se qualifier pour les huitièmes, les quarts, les démies, la finale et le grand vainqueur

ChatGPT m'a sorti un premier code pour vérifier que le fichier était lisible sur Python et pour simuler les matchs. Les simulations de matchs donnaient un résultat aléatoire.

2) **Intégration des Données Historiques :**

J'ai ensuite ajouté un deuxième fichier CSV contenant les résultats des phases finales des dernières années. Ces données historiques ont été utilisées pour calculer un indice de performance pour chaque équipe, basé sur leurs performances passées dans les phases finales. Cela a pour but de rendre la simulation plus réaliste.

> le code fonctionne parfaitement ! j'ai en ma possession un fichier CSV avec les phases finales des dernières années. Est il possible d'ajouter ce fichier pour plus de réalisme sur les prédiction ? En sachant que j'ai uniquement les phases finales donc certaines équipes sont sur le premier fichier mais pas sur le second

Ce fichier a été analysé et préparé après de nombreux problèmes suite aux noms des équipes qui étaient différents du premier fichier (Liverpool / Liverpool FC, Barcelone, FC Barcelona, …)

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/sklearn/base.py:493: UserWarning: X does not have
valid feature names, but RandomForestClassifier was fitted with
feature names
warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/sklearn/base.py:493: UserWarning: X does not have
valid feature names, but RandomForestClassifier was fitted with
feature names
```

Pour résoudre les avertissements "UserWarning: X does not have valid feature names", nous devons nous assurer que les données de prédiction ont les mêmes noms de colonnes que les données d'entraînement.

**3) Calcul de l'Indice de Performance :**

Un score de performance a été attribué à chaque équipe en fonction de leurs performances dans les différentes phases des tournois précédents (huitièmes de finale, quarts de finale, demi-finales, finale, victoire). Cet indice de performance permet de savoir qui sont les favoris et les outsiders. Cela apporte plus de réalisme dans la prédiction car les grands clubs ont plus de chance de remporter la Ligue des Champions.

Les scores ont été pondérés selon la phase atteinte, avec un poids croissant pour les phases plus avancées.

Merci pour le téléchargement du fichier historique. Nous allons maintenant :

1. Charger le fichier historique et le fichier actuel.
2. Calculer un indice de performance basé sur les phases finales historiques.
3. Intégrer cet indice de performance dans notre modèle pour améliorer les prédictions.

**4) Intégration de l'Indice de Performance :**

Les scores de performance ont été associés aux équipes du fichier actuel pour pondérer les probabilités de qualification à chaque phase de la compétition. Cela permet de tenir compte des performances passées pour améliorer les prédictions.

**5) Simulation des Phases de la Compétition :**

**Phase de Groupes** : Les deux premières équipes de chaque groupe se qualifient pour les huitièmes de finale. Les équipes ont été triées par score de performance.

**Huitièmes de Finale, Quarts de Finale, Demi-Finales et Finale** : Chaque match a été simulé en utilisant une fonction qui pondère les résultats en fonction des scores de performance des équipes en compétition.

**Résultats des Simulations**

**Huitièmes de Finale** : Les équipes qualifiées pour les huitièmes de finale ont été déterminées en utilisant les scores de performance et la structure des groupes.

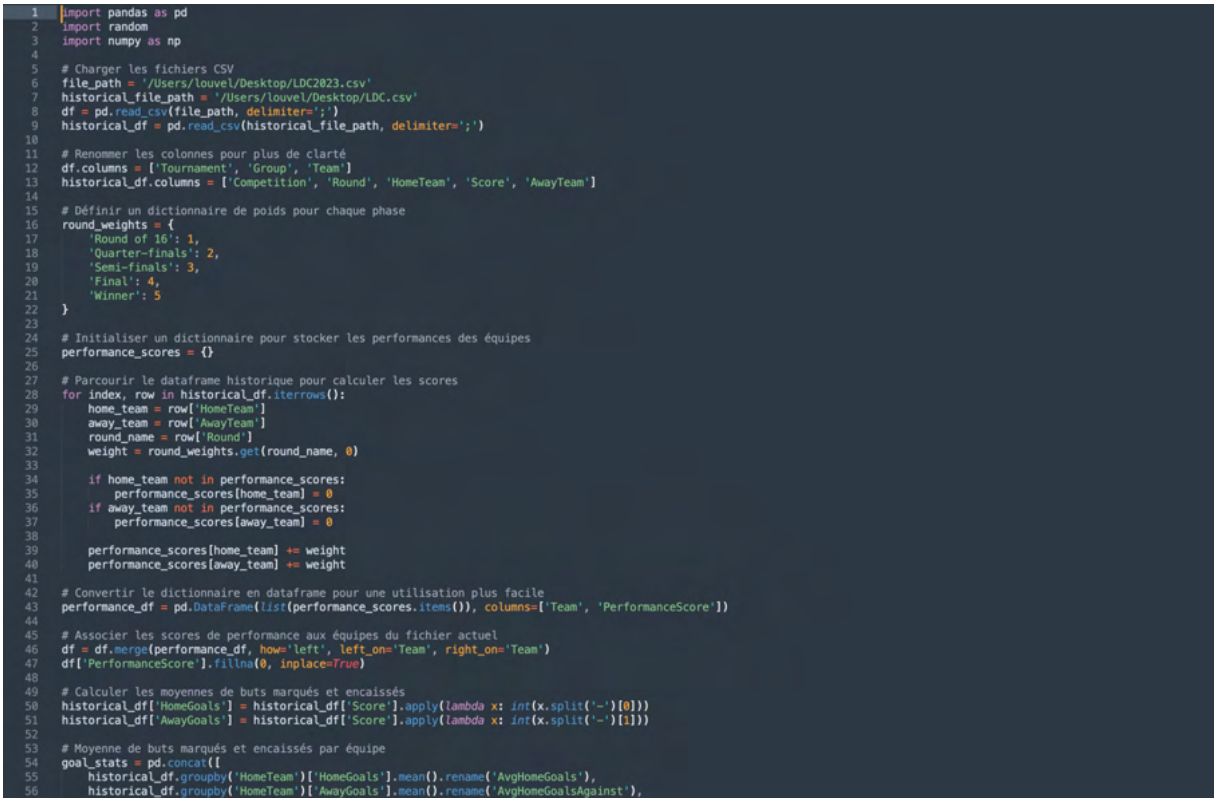
**Quarts de Finale** : Les matchs des huitièmes de finale ont été simulés pour déterminer les équipes qualifiées pour les quarts de finale.

**Demi-Finales** : Les matchs des quarts de finale ont été simulés pour déterminer les équipes qualifiées pour les demi-finales.

**Finale** : Les matchs des demi-finales ont été simulés pour déterminer les équipes qualifiées pour la finale.

**Vainqueur** : Le match final a été simulé pour prédire le vainqueur de la Ligue des Champions 2022/2023.

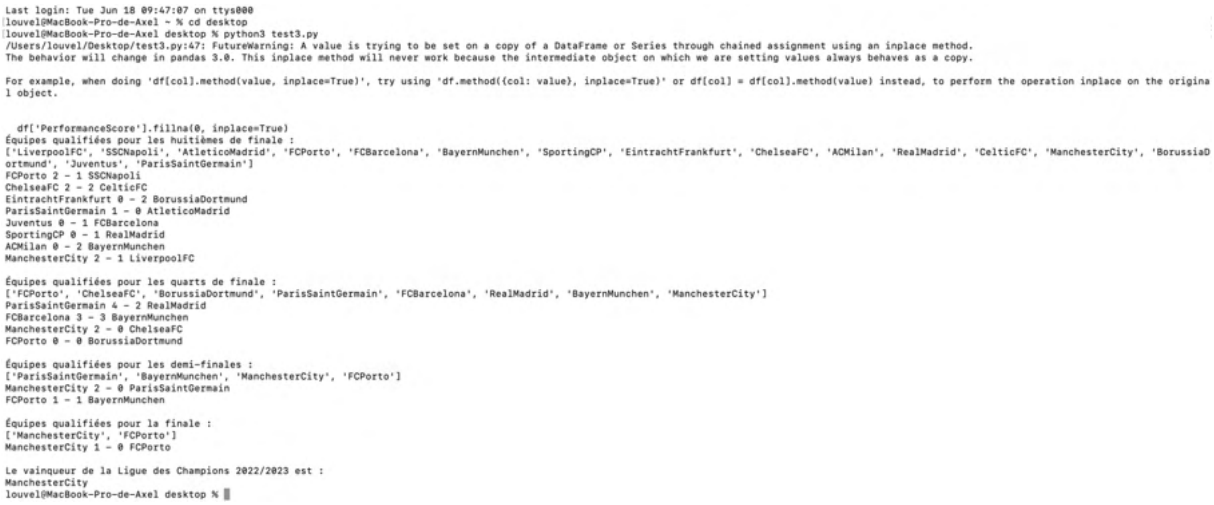
**Code Utilisé :**
Voici le script Python complet utilisé pour réaliser ces simulations :



```python
import pandas as pd
import random
import numpy as np

# Charger les fichiers CSV
file_path = '/Users/louvel/Desktop/LDC2023.csv'
historical_file_path = '/Users/louvel/Desktop/LDC.csv'
df = pd.read_csv(file_path, delimiter=';')
historical_df = pd.read_csv(historical_file_path, delimiter=';')

# Renommer les colonnes pour plus de clarté
df.columns = ['Tournament', 'Group', 'Team']
historical_df.columns = ['Competition', 'Round', 'HomeTeam', 'Score', 'AwayTeam']

# Définir un dictionnaire de poids pour chaque phase
round_weights = {
    'Round of 16': 1,
    'Quarter-finals': 2,
    'Semi-finals': 3,
    'Final': 4,
    'Winner': 5
}

# Initialiser un dictionnaire pour stocker les performances des équipes
performance_scores = {}

# Parcourir le dataframe historique pour calculer les scores
for index, row in historical_df.iterrows():
    home_team = row['HomeTeam']
    away_team = row['AwayTeam']
    round_name = row['Round']
    weight = round_weights.get(round_name, 0)

    if home_team not in performance_scores:
        performance_scores[home_team] = 0
    if away_team not in performance_scores:
        performance_scores[away_team] = 0

    performance_scores[home_team] += weight
    performance_scores[away_team] += weight

# Convertir le dictionnaire en dataframe pour une utilisation plus facile
performance_df = pd.DataFrame(list(performance_scores.items()), columns=['Team', 'PerformanceScore'])

# Associer les scores de performance aux équipes du fichier actuel
df = df.merge(performance_df, how='left', left_on='Team', right_on='Team')
df['PerformanceScore'].fillna(0, inplace=True)

# Calculer les moyennes de buts marqués et encaissés
historical_df['HomeGoals'] = historical_df['Score'].apply(lambda x: int(x.split('-')[0]))
historical_df['AwayGoals'] = historical_df['Score'].apply(lambda x: int(x.split('-')[1]))

# Moyenne de buts marqués et encaissés par équipe
goal_stats = pd.concat([
    historical_df.groupby('HomeTeam')['HomeGoals'].mean().rename('AvgHomeGoals'),
    historical_df.groupby('HomeTeam')['AwayGoals'].mean().rename('AvgHomeGoalsAgainst'),
```

```python
    historical_df.groupby('AwayTeam')['AwayGoals'].mean().rename('AvgAwayGoals'),
    historical_df.groupby('AwayTeam')['HomeGoals'].mean().rename('AvgAwayGoalsAgainst')
], axis=1).fillna(0)

# Fonction pour simuler un score basé sur une distribution de Poisson
def simulate_score(avg_goals):
    return np.random.poisson(avg_goals)

# Simuler la phase de groupes : les deux premières équipes de chaque groupe se qualifient
qualified_teams = []

for group in df['Group'].unique():
    teams_in_group = df[df['Group'] == group][['Team', 'PerformanceScore']].values.tolist()
    teams_in_group.sort(key=lambda x: x[1], reverse=True)  # Trier par score de performance
    # Les deux premières équipes se qualifient
    qualified_teams.extend([team[0] for team in teams_in_group[:2]])

print("Équipes qualifiées pour les huitièmes de finale :")
print(qualified_teams)

# Fonction pour simuler un match avec des scores
def simulate_match_with_scores(team1, team2, goal_stats):
    team1_avg_goals = goal_stats.loc[team1]['AvgHomeGoals'] if team1 in goal_stats.index else 1.2
    team1_avg_goals_against = goal_stats.loc[team1]['AvgHomeGoalsAgainst'] if team1 in goal_stats.index else 1.0
    team2_avg_goals = goal_stats.loc[team2]['AvgAwayGoals'] if team2 in goal_stats.index else 1.1
    team2_avg_goals_against = goal_stats.loc[team2]['AvgAwayGoalsAgainst'] if team2 in goal_stats.index else 1.3

    team1_goals = simulate_score(team1_avg_goals)
    team2_goals = simulate_score(team2_avg_goals)

    print(f"{team1} {team1_goals} - {team2_goals} {team2}")

    if team1_goals > team2_goals:
        return team1
    elif team2_goals > team1_goals:
        return team2
    else:  # En cas d'égalité, choisir aléatoirement
        return random.choice([team1, team2])

# Simuler les huitièmes de finale
random.shuffle(qualified_teams)  # Mélanger les équipes pour simuler les rencontres

# Simuler les gagnants des huitièmes de finale
quarter_final_teams = []

for i in range(0, len(qualified_teams), 2):
    match = qualified_teams[i:i+2]
    winner = simulate_match_with_scores(match[0], match[1], goal_stats)
    quarter_final_teams.append(winner)

print("\nÉquipes qualifiées pour les quarts de finale :")
print(quarter_final_teams)

# Simuler les quarts de finale
random.shuffle(quarter_final_teams)
quarter_final_teams = []

for i in range(0, len(qualified_teams), 2):
    match = qualified_teams[i:i+2]
    winner = simulate_match_with_scores(match[0], match[1], goal_stats)
    quarter_final_teams.append(winner)

print("\nÉquipes qualifiées pour les quarts de finale :")
print(quarter_final_teams)

# Simuler les quarts de finale
random.shuffle(quarter_final_teams)

# Simuler les gagnants des quarts de finale
semi_final_teams = []

for i in range(0, len(quarter_final_teams), 2):
    match = quarter_final_teams[i:i+2]
    winner = simulate_match_with_scores(match[0], match[1], goal_stats)
    semi_final_teams.append(winner)

print("\nÉquipes qualifiées pour les demi-finales :")
print(semi_final_teams)

# Simuler les demi-finales
random.shuffle(semi_final_teams)

# Simuler les gagnants des demi-finales
final_teams = []

for i in range(0, len(semi_final_teams), 2):
    match = semi_final_teams[i:i+2]
    winner = simulate_match_with_scores(match[0], match[1], goal_stats)
    final_teams.append(winner)

print("\nÉquipes qualifiées pour la finale :")
print(final_teams)

# Simuler la finale
winner = simulate_match_with_scores(final_teams[0], final_teams[1], goal_stats)

print("\nLe vainqueur de la Ligue des Champions 2022/2023 est :")
print(winner)
```

**Résultat obtenu après exécution du code :**



```
Last login: Tue Jun 18 09:47:07 on ttys000
louvel@MacBook-Pro-de-Axel ~ % cd desktop
louvel@MacBook-Pro-de-Axel desktop % python3 test3.py
/Users/louvel/Desktop/test3.py:47: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the origina
l object.


  df['PerformanceScore'].fillna(0, inplace=True)
Équipes qualifiées pour les huitièmes de finale :
['LiverpoolFC', 'SSCNapoli', 'AtleticoMadrid', 'FCPorto', 'FCBarcelona', 'BayernMunchen', 'SportingCP', 'EintrachtFrankfurt', 'ChelseaFC', 'ACMilan', 'RealMadrid', 'CelticFC', 'ManchesterCity', 'BorussiaD
ortmund', 'Juventus', 'ParisSaintGermain']
FCPorto 2 - 1 SSCNapoli
ChelseaFC 2 - 2 CelticFC
EintrachtFrankfurt 0 - 2 BorussiaDortmund
ParisSaintGermain 1 - 0 AtleticoMadrid
Juventus 0 - 1 FCBarcelona
SportingCP 0 - 1 RealMadrid
ACMilan 0 - 2 BayernMunchen
ManchesterCity 2 - 1 LiverpoolFC

Équipes qualifiées pour les quarts de finale :
['FCPorto', 'ChelseaFC', 'BorussiaDortmund', 'ParisSaintGermain', 'FCBarcelona', 'RealMadrid', 'BayernMunchen', 'ManchesterCity']
ParisSaintGermain 4 - 2 RealMadrid
FCBarcelona 3 - 3 BayernMunchen
ManchesterCity 2 - 0 ChelseaFC
FCPorto 0 - 0 BorussiaDortmund

Équipes qualifiées pour les demi-finales :
['ParisSaintGermain', 'BayernMunchen', 'ManchesterCity', 'FCPorto']
ManchesterCity 2 - 0 ParisSaintGermain
FCPorto 1 - 1 BayernMunchen

Équipes qualifiées pour la finale :
['ManchesterCity', 'FCPorto']
ManchesterCity 1 - 0 FCPorto

Le vainqueur de la Ligue des Champions 2022/2023 est :
ManchesterCity
louvel@MacBook-Pro-de-Axel desktop %
```

**Conclusion**

Ce projet a permis de mettre en pratique l'utilisation de Python et de ChatGPT, de manipulation de données avec pandas, et de simulation en utilisant des fonctions aléatoires pondérées. L'intégration des données historiques a enrichi le modèle, rendant les prédictions plus réalistes. Ce type de projet peut être étendu en intégrant des modèles de Machine Learning plus sophistiqués pour améliorer encore la précision des prédictions.