Alexandre DEPREZ, Eliot LELEU, Xavier LEDUC, Théo PORZIO

Module Python Scripting

2022 – 2023

1) Introduction

L'objectif de cette analyse de données est de prédire avec un modèle de régression linéaire les points possibles selon le style de jeu des joueurs. Nous verrons qu'ici nous allons étudier les données de tous les joueurs de la NBA les 10 dernières années. Les données sont issues du lien suivant : https://www.nba.com/stats/players/traditional?PerMode=Totals&sort=PTS&dir=-1&Season=....

2) Récupération des données

Afin de récupérer toutes les données des joueurs chaque saison, nous avons donc scraper les données du site en analysant la table qui est donnée avec la librairie BeautifulSoup :

```python
def get_advanced_data():
    i = 0
    for year in seasons:
        link = "https://www.nba.com/stats/players/traditional?PerMode=Totals&sort=PTS&dir=-1&Season={}".format(seasons[i])
        driver = webdriver.Chrome()
        driver.get(link)

        cookie_accept = driver.find_element(By.CSS_SELECTOR, "#onetrust-accept-btn-handler")
        cookie_accept.click()
        time.sleep(10)

        dropdown_div = driver.find_element(By.CSS_SELECTOR, ".Pagination_pageDropdown__KgjBU")
        dropdown_element = Select(dropdown_div.find_element(By.CSS_SELECTOR, ".DropDown_select__4pIg9"))
        dropdown_element.select_by_visible_text("All")

        page_source = driver.page_source

        soup = BeautifulSoup(page_source, 'html.parser')

        table = soup.find('table', attrs={'class':'Crom_table__p1iZz'})

        if table:
            df = pd.read_html(io.StringIO(str(table)))[0]

            df.to_csv('NBA_Stats_Advanced_{}.csv'.format(seasons[i]), index=False)


            print("Données sauvegardées.")
        else:
            print("Erreur lors de la récupération des données.")

        time.sleep(10)
        i += 1
```

Ainsi le programme va donc récupérer toutes les données des joueurs par année. Le principe est simple, nous souhaitons analyser les dix dernières années, alors faire une boucle for est la méthode la plus juste. Chaque saison est enregistrée dans un fichier csv afin de faire des tests intermédiaires afin d'être sûr que le modèle de prédilection soit précis.

Maintenant il faut assembler toutes ces saisons sur un seul fichier CSV pour avoir beaucoup de données à analyser.

```
def assemble_all_seasons_advanced():
    i = 0
    for year in seasons:
        df = pd.read_csv('NBA_Stats_Advanced_{}.csv'.format(seasons[i]))
        df['SEASON'] = seasons[i]
        if i == 0:
            df_all = df
        else:
            df_all = pd.concat([df_all, df])
        i += 1
    df_all.to_csv('NBA_Stats_Advanced_All_Seasons.csv', index=False)
```

Ici cette fonction nous permettra de stocker toutes les données dans un seul fichier CSV. Cependant, nous allons retrouver des joueurs en plusieurs, alors nous allons donc moyenner les données dans un seul fichier.

```
def assembly_by_player(df):
    df = df.drop(['Unnamed: 0', 'Team', 'GP RANK', 'W RANK', 'L RANK', 'MIN RANK', 'PTS RANK', 'FGM RANK', 'FGA RANK',
                  'FG% RANK', '3PM RANK', '3PA RANK', '3P% RANK', 'FTM RANK', 'FTA RANK', 'FT% RANK', 'OREB RANK', 'DREB RANK',
                  'REB RANK', 'AST RANK', 'TOV RANK', 'STL RANK', 'BLK RANK', 'PF RANK', 'FP RANK', 'DD2 RANK', 'TD3 RANK', '+/- RANK', 'SEASON'],
                  axis=1)

    df_avg = df.groupby('Player').mean().reset_index()

    df_avg.to_csv('NBA_Stats_Advanced_Group_By_Player_All_Season.csv', index=False)
```

Cette fonction nous donne la possibilité de rassembler les données d'un seul joueur des dix dernières, maintenant s'il le joueur était présent sur plusieurs années, nous avons choisis de faire la moyenne de ses données, ainsi nous avons une pondération de ses données sur ces années. Seulement, un autre problème se pose devant nous, certains joueurs n'ont pas participé ou ont peu de données à traiter pour le modèle, le prepossecing est donc important. Ainsi, dans la fonction « assembly_by_players » retire les colonnes non essentielles.

3) Modèle de classification

Nous avons utilisé la librairie sklearn pour classifier nos joueurs. Afin que ce modèle soit juste, nous avons choisis de prendre les les 25%, 50% et 75% des points de tous les joueurs. Ainsi, cette fonction permet de classifier les joueurs :

```python
def classification_joueurs(df):
    print(df['PTS'].describe())
    df['Level'] = pd.cut(df['PTS'], bins=[46, 208, 488, float('inf')], labels=['Faible', 'Moyen', 'Élevé'])

    features = df[['FGM', '3PM', 'FTM', 'OREB', 'DREB', 'AST', 'STL', 'BLK', 'TOV', 'PF']]
    target = df['Level']

    label_encoder = LabelEncoder()
    target = label_encoder.fit_transform(target)

    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(features)

    X_train, X_test, y_train, y_test = train_test_split(scaled_features, target, test_size=0.3, random_state=42)

    n_neighbors = 3
    knn_model = KNeighborsClassifier(n_neighbors=n_neighbors)

    knn_model.fit(X_train, y_train)

    predictions = knn_model.predict(X_test)

    accuracy = accuracy_score(y_test, predictions)
    classification_report_output = classification_report(y_test, predictions)

    print("Accuracy:", accuracy)
    print("Classification Report:\n", classification_report_output)


    selected_columns = ['Player', 'PTS', 'Level']
    df_selected = df[selected_columns]
    df_selected.to_csv('Classification_Players.csv', index=False)

    print("Fichier CSV enregistré avec succès.")
```

Dans cette fonction, à la ligne où se trouve bins, se trouve les pourcentages des points selon le df.describe(). Grâce à cela nous déterminons donc les joueurs qui seront dans les 3 groupes : Faible, Moyen, Elevé.

Pour vérifier notre modèle, nous avons afficher la précision et d'autres paramètres:

```
Name: PTS, dtype: float64
Accuracy: 0.8818181818181818
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.82      0.86       112
           1       0.80      0.85      0.83       114
           2       0.91      0.90      0.90       117
           3       0.92      0.97      0.94        97

    accuracy                           0.88       440
   macro avg       0.88      0.88      0.88       440
weighted avg       0.88      0.88      0.88       440
```

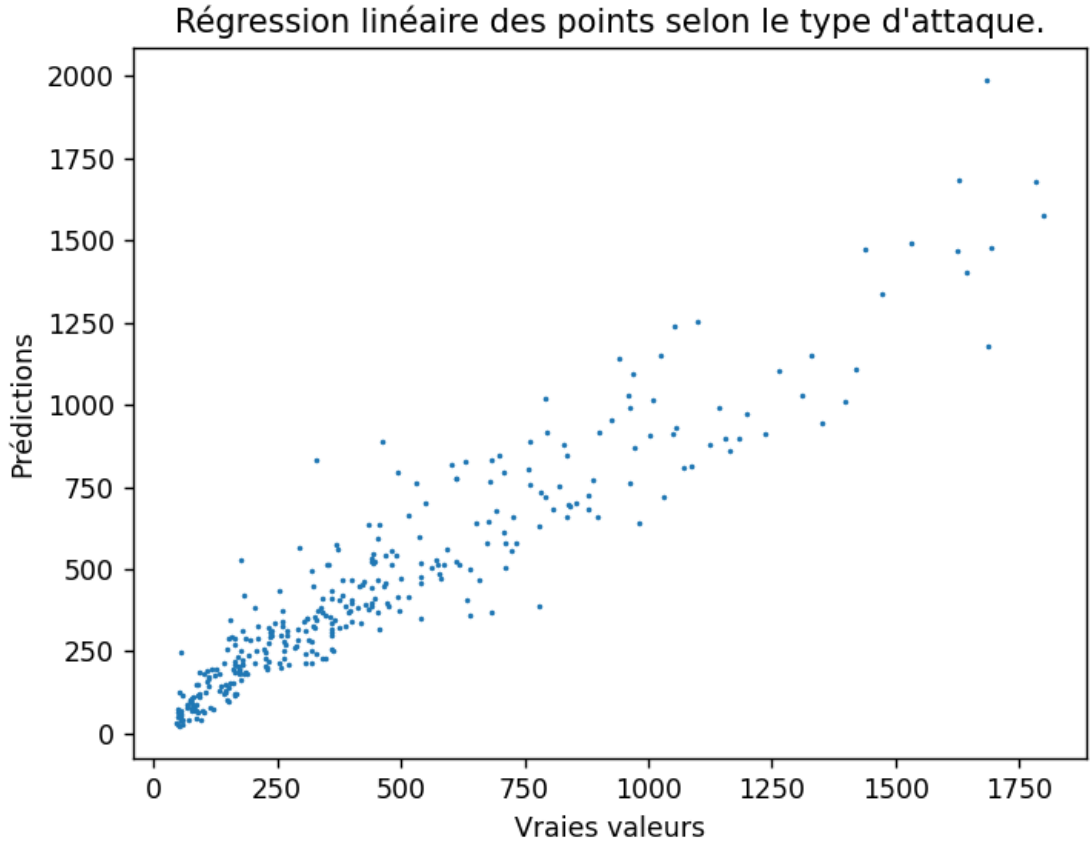
A savoir que tous nos modèles ont été entrainés sur 70% des données, et testés sur 30% des données.

De plus, une vérification humaine était importante. Selon deux sites suivants l'actualité du basketball américain, nous ont permis de confirmer que dans le fichier CSV avec la classification des joueurs que ces données étaient justes.

4) Modèle de prédilection

Maintenant, que nous avions classifié les joueurs, le plus important de ce sujet est donc de faire une régression linéaire avec les attaques des joueurs.

Voici notre résultat :



Ici nous pouvons voir notre modèle de régression linéaire.

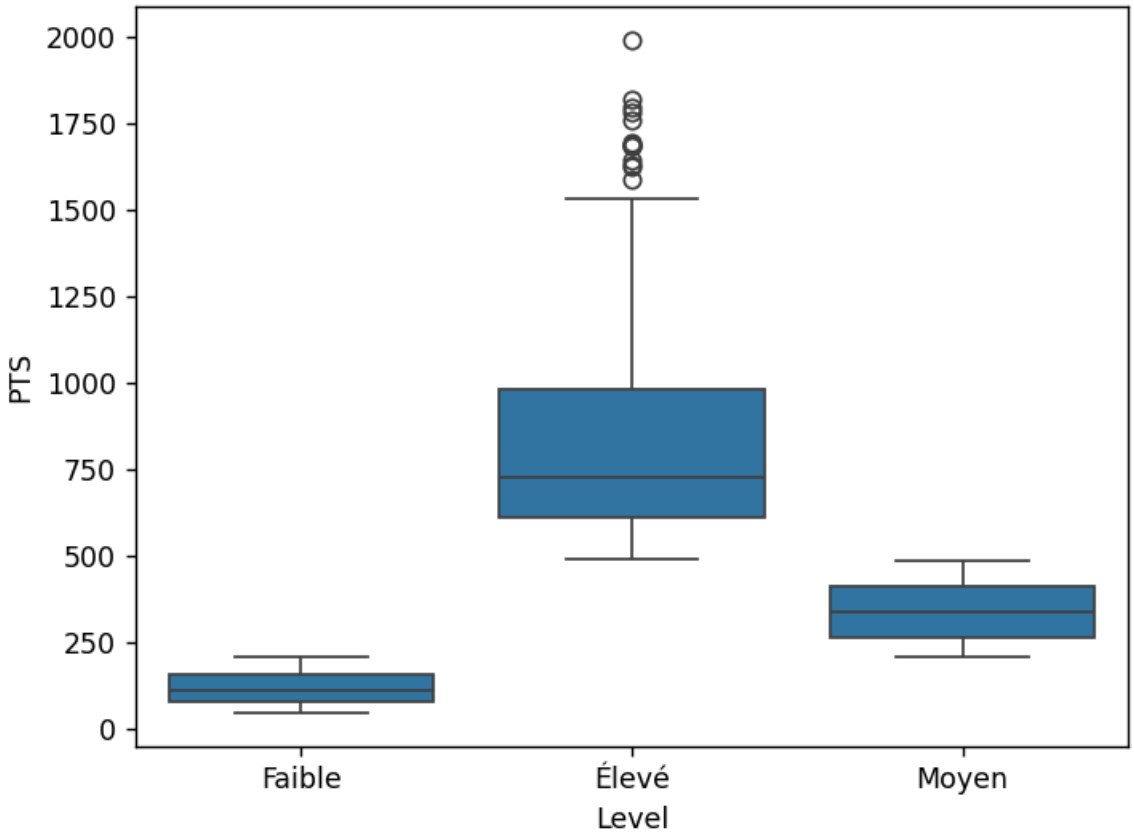
Ce modèle a eu comme objectif les points selon les actions du joueur, par ex : rebonds offensifs, défensifs, etc.

A savoir que nous avons vérifié ce modèle :

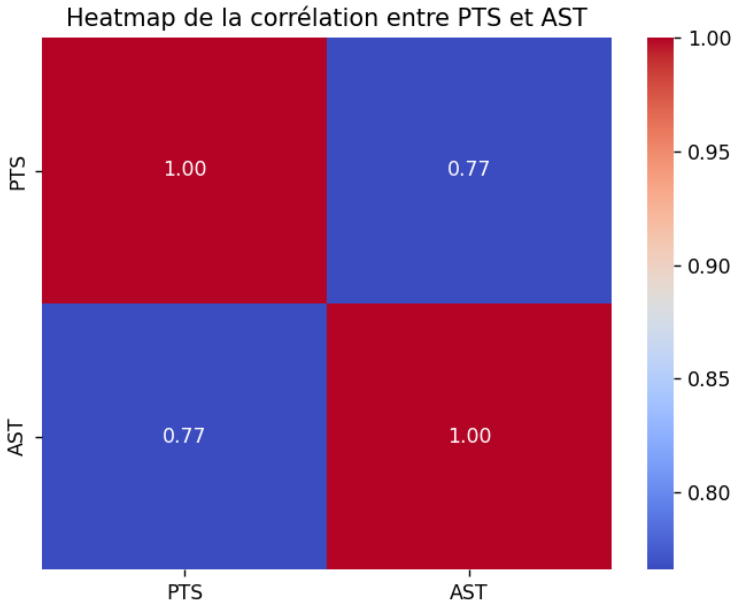
```
Mean Squared Error: 16250.898260863436
R-squared: 0.8876862325158136
```

5) Graphiques des données

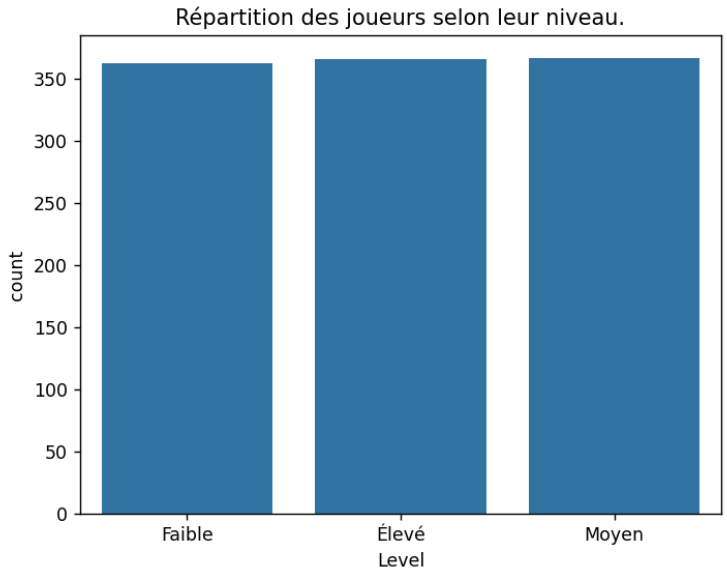
Nous avons affiché un boxplot de la classification pour montrer les écarts avec notre modèle :



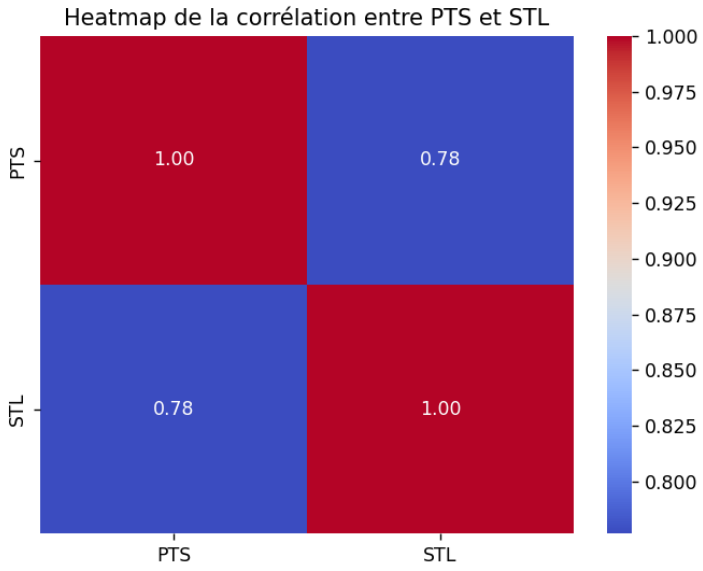
Dans la catégorie des joueurs élevés, il y a beaucoup de données en dehors de ce champ, ainsi un ajustement de la classification devra être fait.



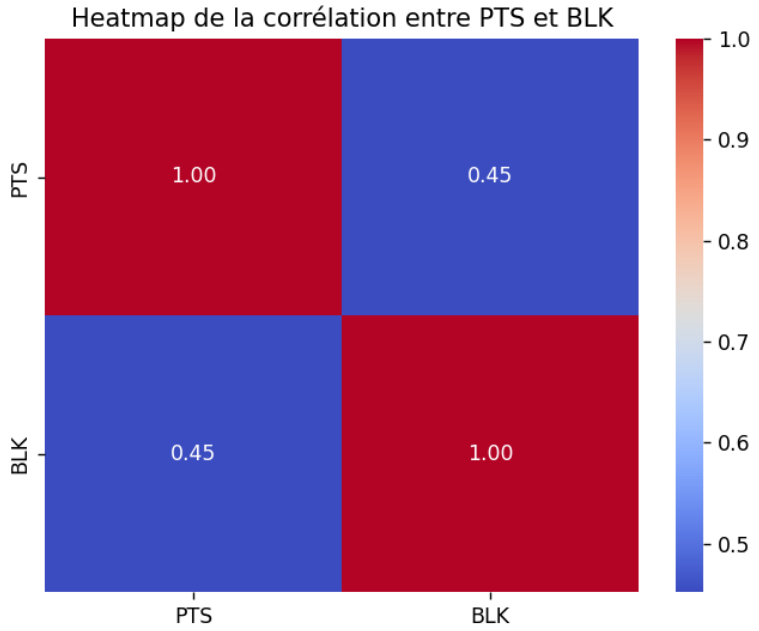
Un histogramme de la répartition des joueurs selon leur niveau :



Une heatmap montrant la corrélation avec les ballons volés :



Et enfin, une heatmap des points avec la corrélation des ballons bloqués :



6) Difficultés rencontrées :
   La principale difficulté était le traitement des données, en effet, les tableaux comportés des données vides, intraitables pour nos modèles.

7) Finalité :
   Nous avons su analyser les données joueurs pour prédire les points avec leur style d'attaque.