Report on NFL Fantasy Football Dataset & ML Model

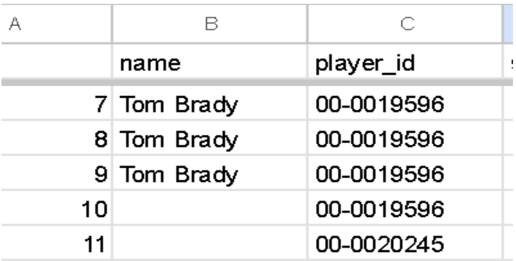
By: Mukul Garg

Data Analysis

The provided dataset contains historical statistical data of players. The task was to analyze the data and build an ML model that accurately predicts the number of points each NFL player will score in the upcoming season.

Let us start with data analysis.

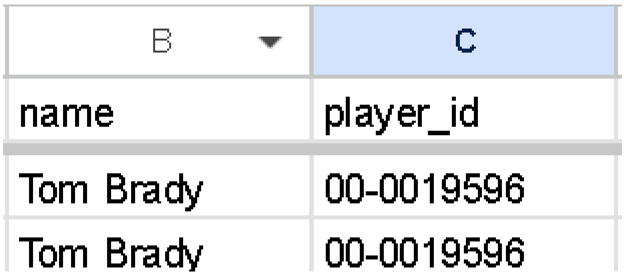
On having a first look at the dataset, we can observe multiple columns containing information about players are provided but there are many empty cells in some columns. The first such column is 'name' which as the name suggests contains names of players. Dataset has a total of 4655 entries and out of those, 1057 cells in 'name' column are empty. On further observation it was noted that the names of players are repeated multiple times for different seasons but for the same player there were instances where name was not provided. This could be easily inferred as every row had been provided a player id which is unique for every player.



A	B	C
	name	player_id
7	Tom Brady	00-0019596
8	Tom Brady	00-0019596
9	Tom Brady	00-0019596
10		00-0019596
11		00-0020245

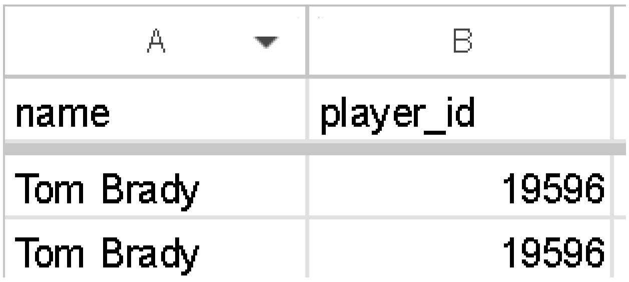
Out of 1057 empty values, there are 290 players whose name does not appear at all in the dataset. For players whose name appeared even once in the dataset, I used python program to fill correct names in respective empty spaces.

Next column is 'player_id' but player ids were given in an absurd format which could have created complexities, so I changed the format from '00-00NNNNN' to just 'NNNNN'



B	C
name	player_id
Tom Brady	00-0019596
Tom Brady	00-0019596

to



A	B
name	player_id
Tom Brady	19596
Tom Brady	19596

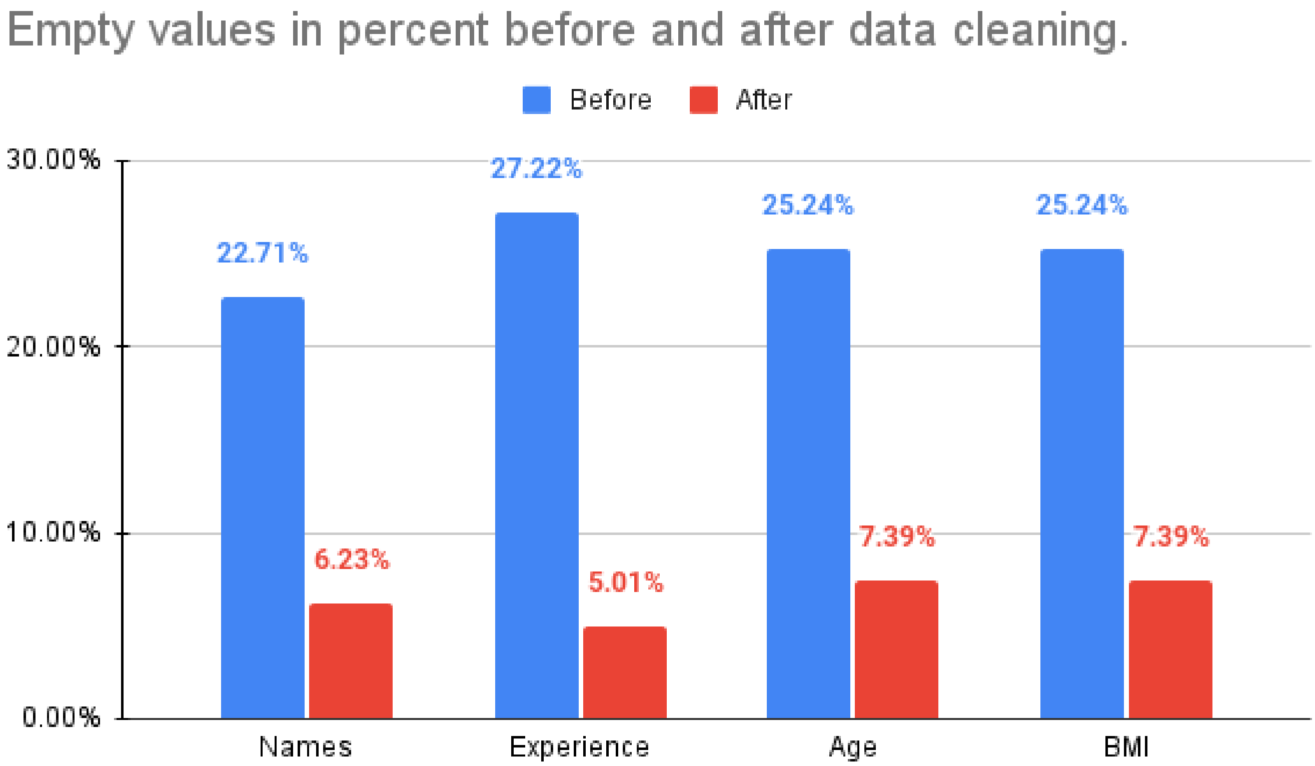
.

It was observed that 1614 unique players exist in the dataset.

Next columns having missing values are ‘years_exp’ which contains experience in years of players and ‘age’ which contains data of player’s age in years.

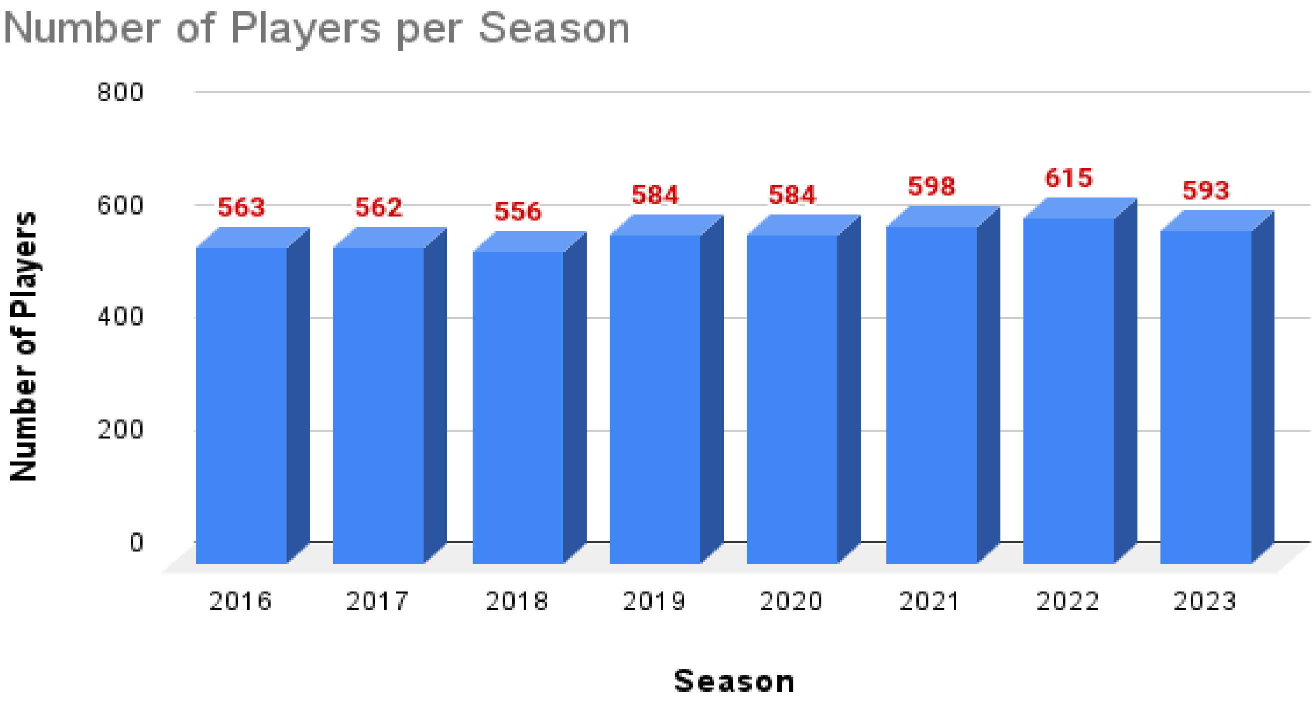
It was observed that a total of 1267 cells were empty in ‘years_exp’ column and 1175 empty cells for ‘age’. Experience for players who were playing for the first time in 2023 was not listed and for many other players too who have played in the past. For e.g., a player who played in 2019 and had an experience of 5 years, had empty cell in ‘years_exp’ column for season 2020 which should have been 6 years. So, I again used a python program to fill empty columns for all those who were playing for the first time as they clearly had 0 years of experience and calculated missing values of experience for players whose past data was available in the dataset depending on what their experience was in the last season they played. Similarly, also calculated age for players. After data cleaning, only 286 empty values for experience and 344 empty values for age column are there.

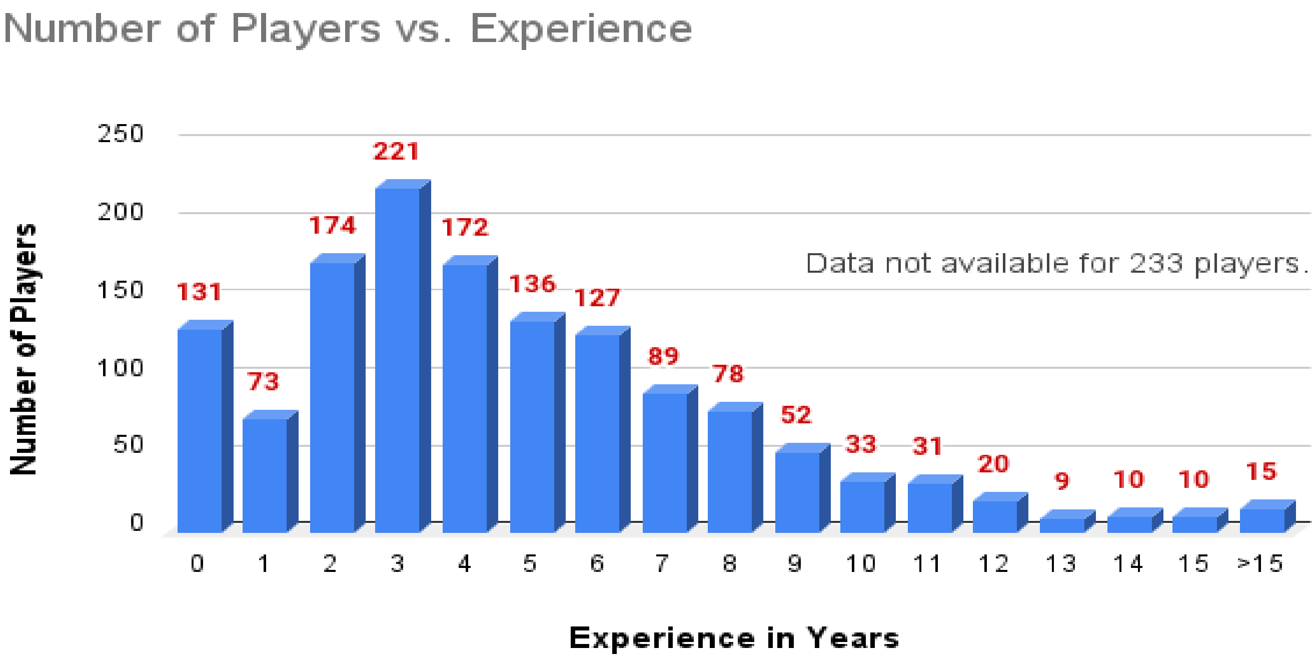
BMI was not provided for 1175 rows, but it was same for a player throughout the seasons, so it was also easy to clean the BMI column. If BMI for a player is given even once, it remains the same for all the seasons in which he played so again I used a python program to clean this column. Only 344 values were empty after data cleaning.

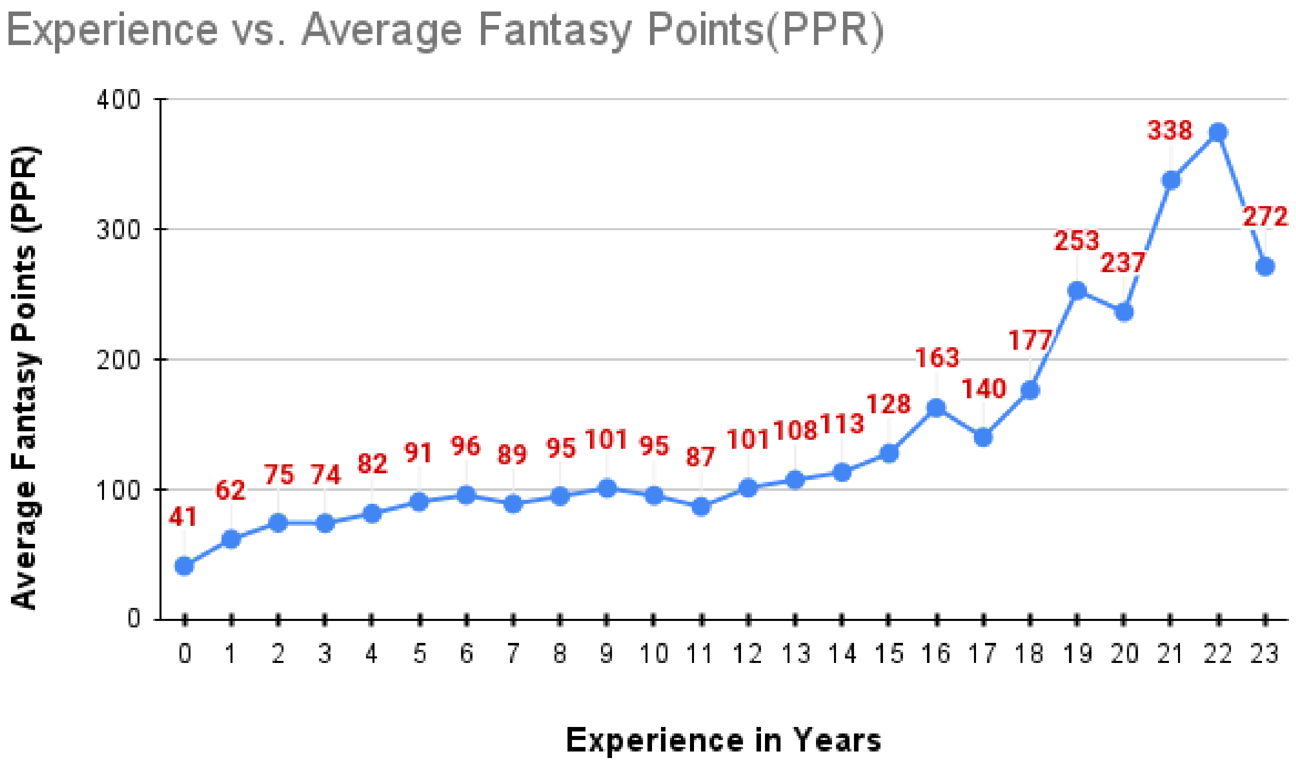

Empty values in percent before and after data cleaning.
Before
After
30.00%
20.00%
10.00%
0.00%
22.71%
6.23%
27.22%
5.01%
25.24%
7.39%
25.24%
7.39%
Names
Experience
Age
BMI

Data Interpretation and Patterns

After basic data cleaning, I tried to interpret data and find patterns. Some of them are :

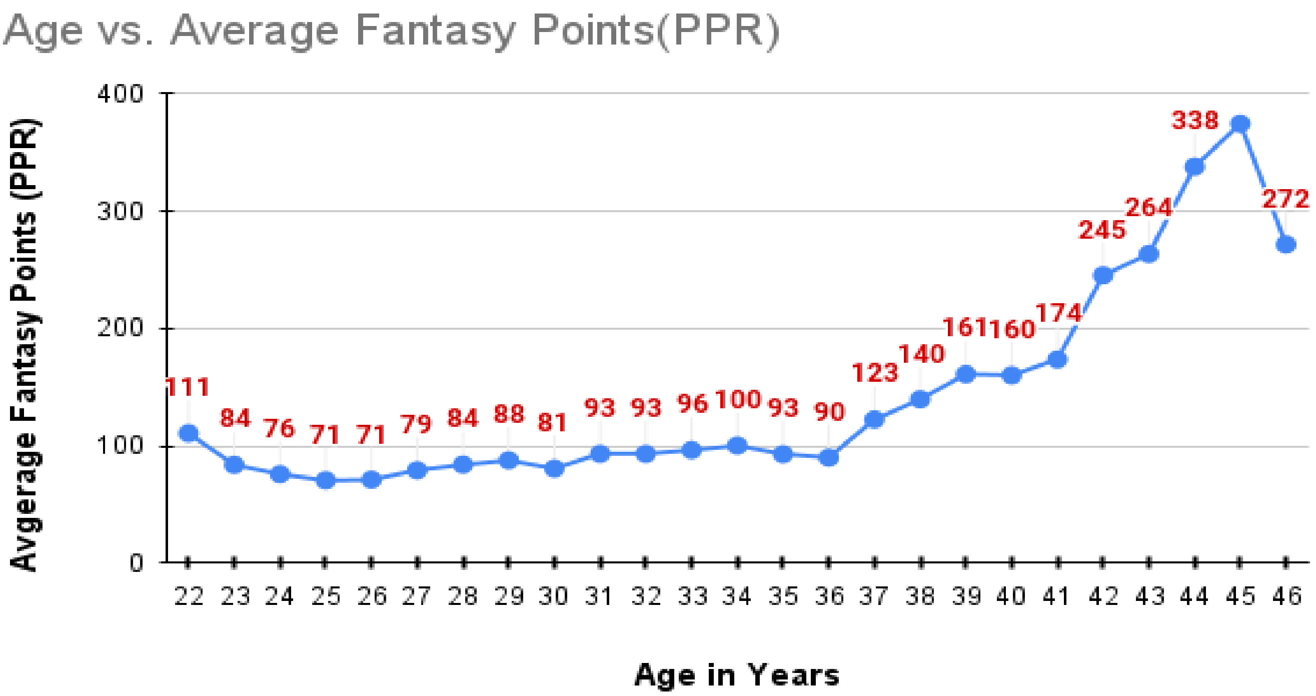




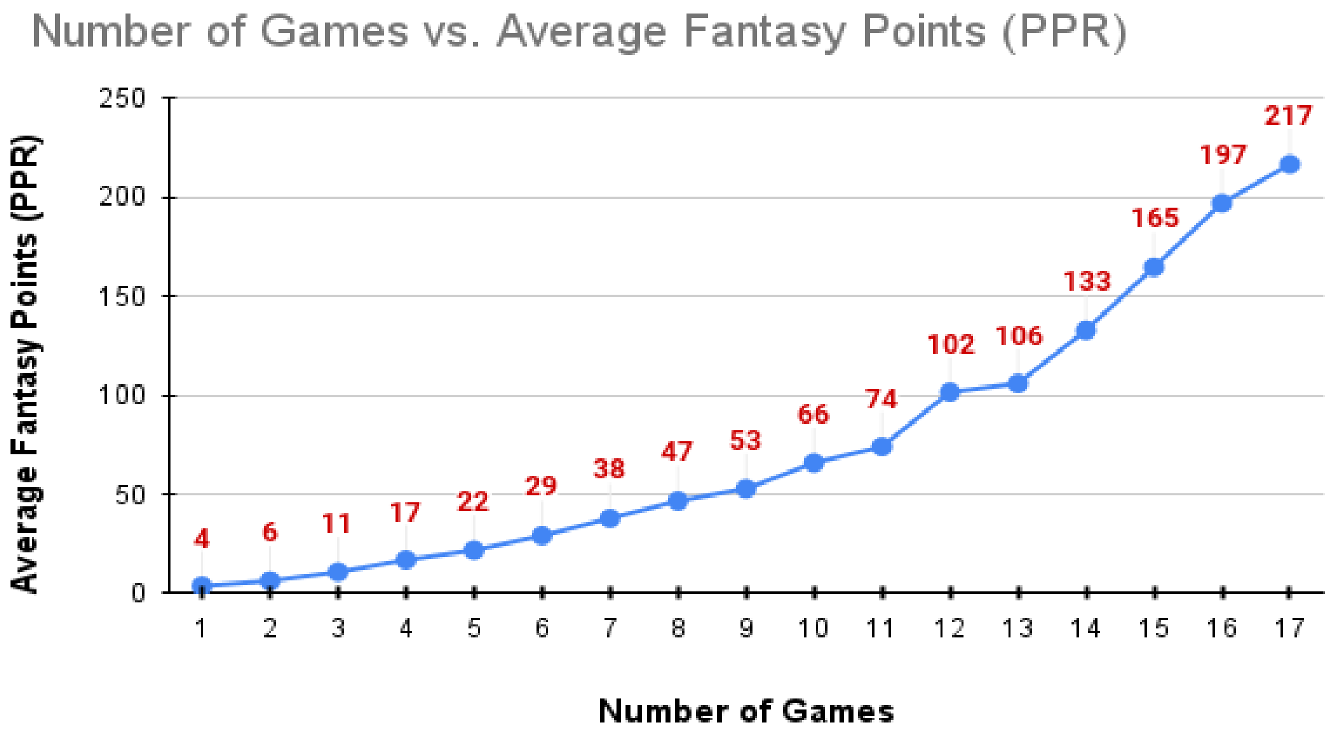


It can be observed that Average fantasy points is in uptrend as the experience increases with a significant drop only in the last which could be because of the effect of age stars kicking in.

Similar trend can be observed with age, but uptrend of fantasy points starts only after the age of 37 years.

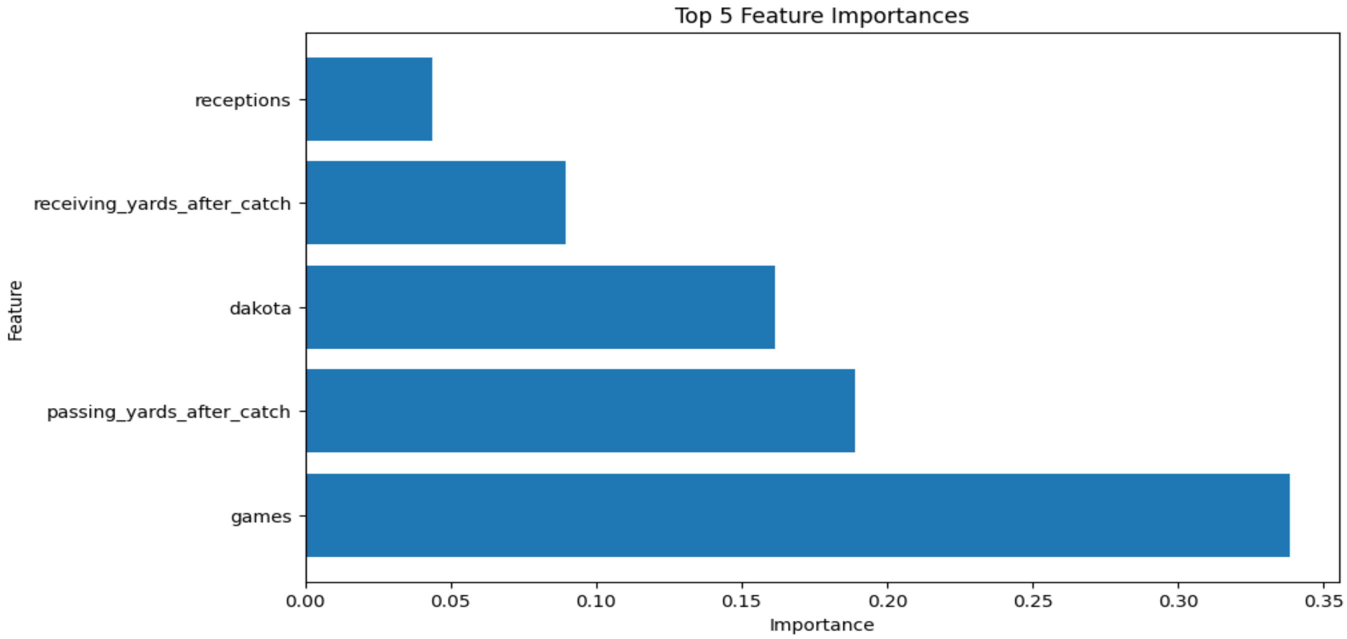


One of the best patterns is observed to be Number of games played vs Average Fantasy Points.



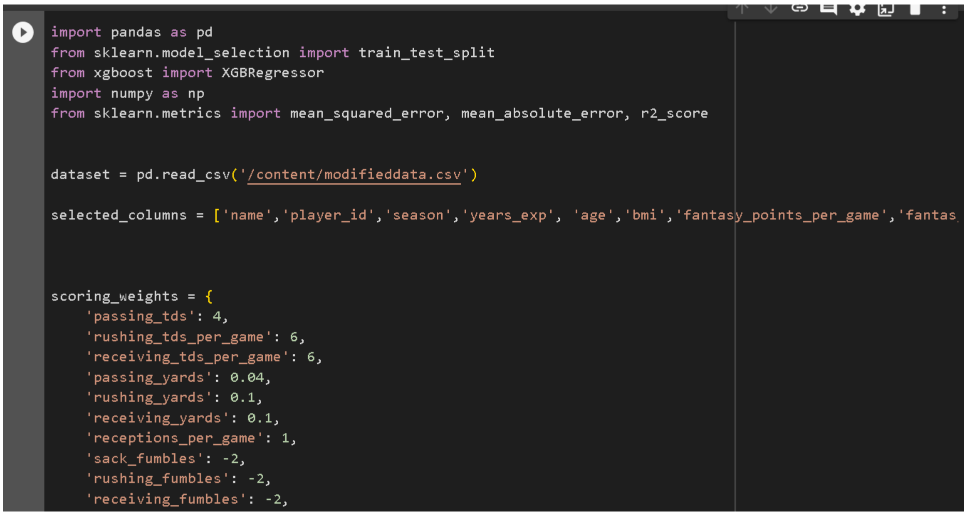
This is also evident in the importance of features chart which I generated using python and plotted in matplot lib.

https://colab.research.google.com/drive/1zWgzf_AYCl0RTKVA8ksCzLHixvzmml80?usp=sharing

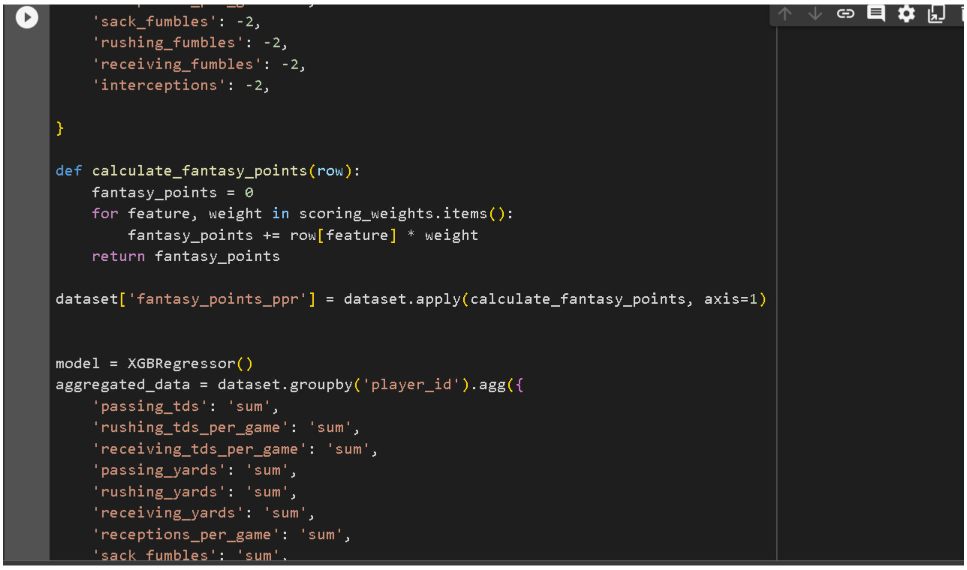


Model Selection

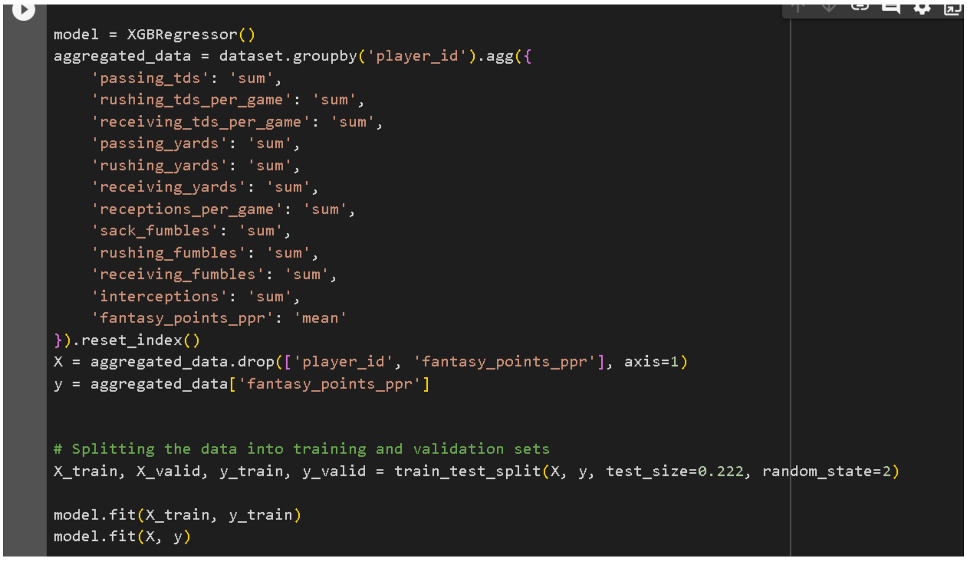
For this dataset, I am going to use XGBRegressor simply because it can handle complex relations, provide feature importance, handle missing values and is a high-performance ML model.



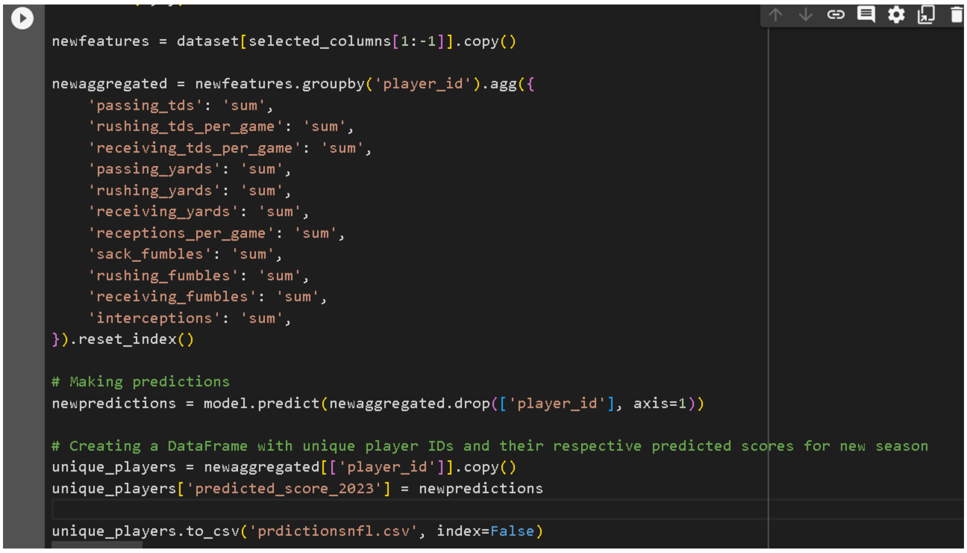
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


dataset = pd.read_csv('/content/modifieddata.csv')

selected_columns = ['name','player_id','season','years_exp', 'age','bmi','fantasy_points_per_game','fantas



scoring_weights = {
    'passing_tds': 4,
    'rushing_tds_per_game': 6,
    'receiving_tds_per_game': 6,
    'passing_yards': 0.04,
    'rushing_yards': 0.1,
    'receiving_yards': 0.1,
    'receptions_per_game': 1,
    'sack_fumbles': -2,
    'rushing_fumbles': -2,
    'receiving_fumbles': -2,
```



```python
    'sack_fumbles': -2,
    'rushing_fumbles': -2,
    'receiving_fumbles': -2,
    'interceptions': -2,

}

def calculate_fantasy_points(row):
    fantasy_points = 0
    for feature, weight in scoring_weights.items():
        fantasy_points += row[feature] * weight
    return fantasy_points

dataset['fantasy_points_ppr'] = dataset.apply(calculate_fantasy_points, axis=1)


model = XGBRegressor()
aggregated_data = dataset.groupby('player_id').agg({
    'passing_tds': 'sum',
    'rushing_tds_per_game': 'sum',
    'receiving_tds_per_game': 'sum',
    'passing_yards': 'sum',
    'rushing_yards': 'sum',
    'receiving_yards': 'sum',
    'receptions_per_game': 'sum',
    'sack_fumbles': 'sum',
```



```
model = XGBRegressor()
aggregated_data = dataset.groupby('player_id').agg({
    'passing_tds': 'sum',
    'rushing_tds_per_game': 'sum',
    'receiving_tds_per_game': 'sum',
    'passing_yards': 'sum',
    'rushing_yards': 'sum',
    'receiving_yards': 'sum',
    'receptions_per_game': 'sum',
    'sack_fumbles': 'sum',
    'rushing_fumbles': 'sum',
    'receiving_fumbles': 'sum',
    'interceptions': 'sum',
    'fantasy_points_ppr': 'mean'
}).reset_index()
X = aggregated_data.drop(['player_id', 'fantasy_points_ppr'], axis=1)
y = aggregated_data['fantasy_points_ppr']


# Splitting the data into training and validation sets
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.222, random_state=2)

model.fit(X_train, y_train)
model.fit(X, y)
```

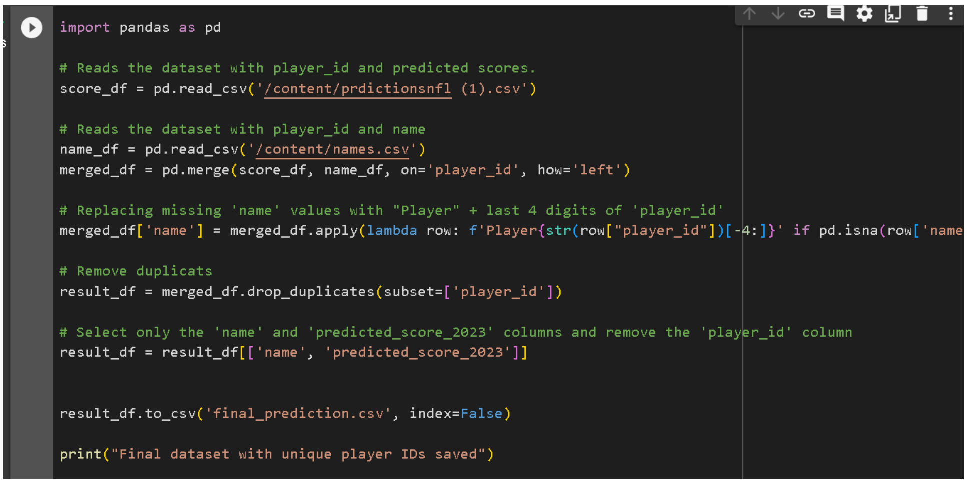


```
newfeatures = dataset[selected_columns[1:-1]].copy()

newaggregated = newfeatures.groupby('player_id').agg({
    'passing_tds': 'sum',
    'rushing_tds_per_game': 'sum',
    'receiving_tds_per_game': 'sum',
    'passing_yards': 'sum',
    'rushing_yards': 'sum',
    'receiving_yards': 'sum',
    'receptions_per_game': 'sum',
    'sack_fumbles': 'sum',
    'rushing_fumbles': 'sum',
    'receiving_fumbles': 'sum',
    'interceptions': 'sum',
}).reset_index()

# Making predictions
newpredictions = model.predict(newaggregated.drop(['player_id'], axis=1))

# Creating a DataFrame with unique player IDs and their respective predicted scores for new season
unique_players = newaggregated[['player_id']].copy()
unique_players['predicted_score_2023'] = newpredictions

unique_players.to_csv('prdictionsnfl.csv', index=False)
```

These are the snapshots of model used.

https://colab.research.google.com/drive/1PiOZQleUwfVyFPM4u0PQbFV3u6PRKH1m?usp=sharing

This provides output as csv file having player id and respective predicted score. Then I used another code to replace player id with correct names and for players whose names were not mentioned in the dataset at all, I am using PlayerNNNN where NNNN is last 4 digits from their player id.

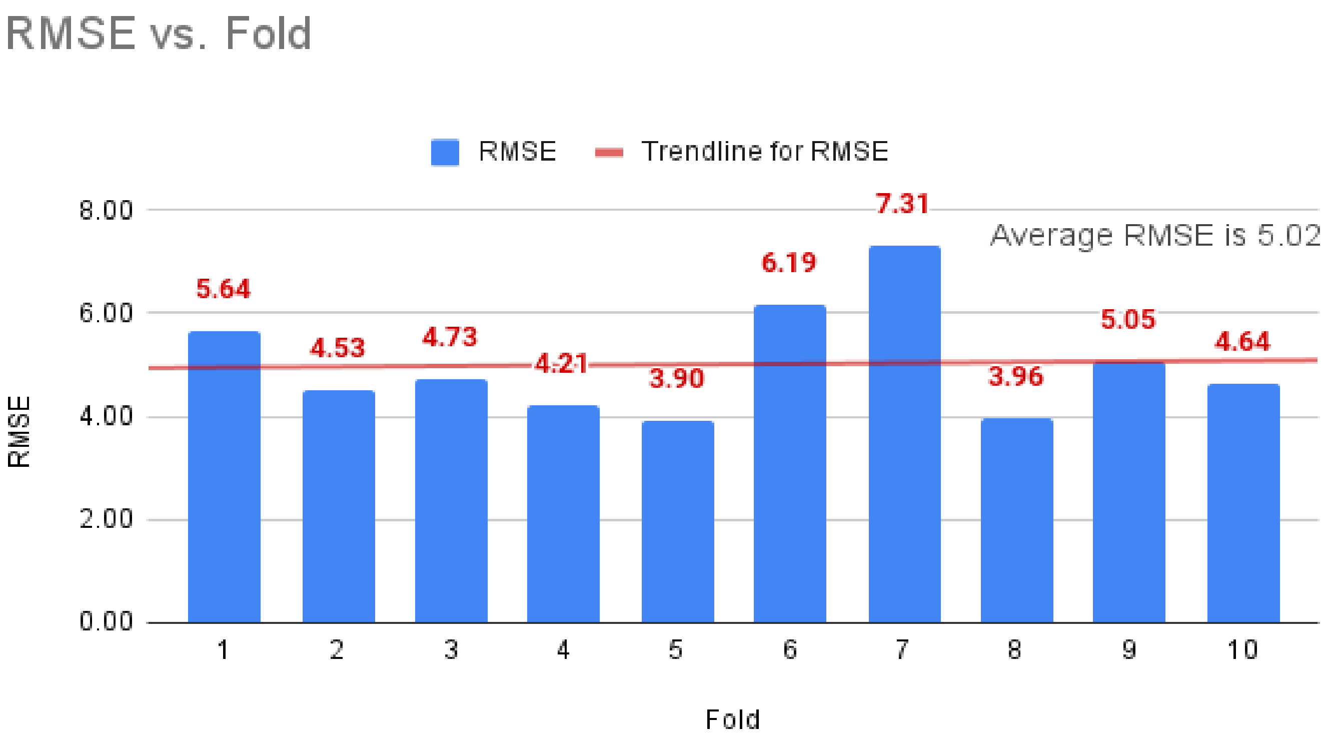


```
import pandas as pd

# Reads the dataset with player_id and predicted scores.
score_df = pd.read_csv('/content/prdictionsnfl (1).csv')

# Reads the dataset with player_id and name
name_df = pd.read_csv('/content/names.csv')
merged_df = pd.merge(score_df, name_df, on='player_id', how='left')

# Replacing missing 'name' values with "Player" + last 4 digits of 'player_id'
merged_df['name'] = merged_df.apply(lambda row: f'Player{str(row["player_id"])[-4:]}' if pd.isna(row['name

# Remove duplicats
result_df = merged_df.drop_duplicates(subset=['player_id'])

# Select only the 'name' and 'predicted_score_2023' columns and remove the 'player_id' column
result_df = result_df[['name', 'predicted_score_2023']]


result_df.to_csv('final_prediction.csv', index=False)

print("Final dataset with unique player IDs saved")
```

https://colab.research.google.com/drive/15kXG36AzxzZwJM_hJpULroW_CucFAIWi?usp=sharing

Model Evaluation

Basic parameters were calculated for evaluation.



https://colab.research.google.com/drive/1ROOp920qeRsMyyTNHjn4c47WMcvzCB47?usp=sharing

Google Colab links of all the python programs used:

- For handling missing values of experience, age and bmi: https://colab.research.google.com/drive/1WU2vyzVZZPn064r9vNmPETcPw_rTDSX5?usp=sharing
- Feature importance : https://colab.research.google.com/drive/1zWgzf_AYCl0RTKVA8ksCzLHixvzmml80?usp=sharing
- Model : https://colab.research.google.com/drive/1PiOZQleUwfVyFPM4u0PQbFV3u6PRKH1m?usp=sharing
- Replacing playerid with names and handling players without name in dataset: https://colab.research.google.com/drive/15kXG36AzxzZwJM_hJpULroW_CucFAIWi?usp=sharing
- Model evaluation : https://colab.research.google.com/drive/1ROOp920qeRsMyyTNHjn4c47WMcvzCB47?usp=sharing

All the charts used in this report are made by me. Graphics on cover page are taken from : https://www.playitusa.com/wp-content/uploads/2020/02/nfl-fffffffff.png