Fantasy_Football_ML

October 14, 2024

[1]:
```python
# Imports
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
import numpy as np
import re
```

[2]:
```python
# Week 2 Start: Ingestion of the Dataset

# File paths for each year
file_2019 = 'FFRank 2019.csv'
file_2020 = 'FFRank 2020.csv'
file_2021 = 'FFRank 2021.csv'
file_2022 = 'FFRank 2022.csv'
```

[3]:
```python
# Read each CSV file into a pandas DataFrame
df_2019 = pd.read_csv(file_2019)
df_2020 = pd.read_csv(file_2020)
df_2021 = pd.read_csv(file_2021)
df_2022 = pd.read_csv(file_2022)
```

[4]:
```python
# Display the first few rows of each dataset to ensure they loaded correctly
print("2019 Data Preview:\n", df_2019.head())
print("2020 Data Preview:\n", df_2020.head())
print("2021 Data Preview:\n", df_2021.head())
print("2022 Data Preview:\n", df_2022.head())
```

```
2019 Data Preview:
    Rank              Player Team Position  Age  Games Played  \
0     1  Christian McCaffrey  CAR       RB   23            16
1     2         Lamar Jackson  BAL       QB   22            15
2     3        Derrick Henry  TEN       RB   25            15
```

```
3     4          Aaron Jones  GNB       RB   25            16
4     5      Ezekiel Elliott  DAL       RB   24            16

   Passing Completion  Passing Attempts  Passing Yards   Passing TDs  …  \
0                   0                 2              0             0  …
1                 265               401           3127            36  …
2                   0                 0              0             0  …
3                   0                 0              0             0  …
4                   0                 0              0             0  …

   Rushing TDs  Targets  Recepotions  Receiving Yards  Yards per Reception  \
0           15      142          116             1005                 8.66
1            7        0            0                0                  NaN
2           16       24           18              206                11.44
3           16       68           49              474                 9.67
4           12       71           54              420                 7.78

   Receiving TDs  Fumbles Lost  Total TD  Fantasy Points  PPR Fantasy Points
0              4             0        19             355               471.2
1              0             2         7             416               415.7
2              2             3        18             277               294.6
3              3             2        19             266               314.8
4              2             2        14             258               311.7

[5 rows x 24 columns]
2020 Data Preview:
    Rank          Player Team Position  Age  Games Played  Passing Completions  \
0      1  Derrick Henry  TEN       RB   26            16                    0
1      2   Alvin Kamara  NOR       RB   25            15                    0
2      3    Dalvin Cook  MIN       RB   25            14                    0
3      4  Davante Adams  GNB       WR   28            14                    0
4      5   Travis Kelce  KAN       TE   31            15                    1

   Passing Attempts  Passing Yards  Passing TD  …  Rushing TD  Targets   \
0                 0              0           0  …          17       31
1                 0              0           0  …          16      107
2                 0              0           0  …          16       54
3                 0              0           0  …           0      149
4                 2              4           0  …           0      145

   Receptions  Receiving Yards  Yards Per Reception  Receiving TD  \
0          19              114                 6.00             0
1          83              756                 9.11             5
2          44              361                 8.20             1
3         115             1374                11.95            18
4         105             1416                13.49            11

   Fumles Lost  Total TD  Fantasy Points  PPR Points
```

```
0            2        17             314       333.1
1            0        21             295       377.8
2            3        17             294       337.8
3            1        18             243       358.4
4            1        11             208       312.8

[5 rows x 24 columns]
2021 Data Preview:
   Rank           Player Team Position  Age  Games Played  \
0     1  Jonathan Taylor  IND       RB   22            17
1     2      Cooper Kupp  LAR       WR   28            17
2     3     Deebo Samuel  SFO       WR   25            16
3     4       Josh Allen  BUF       QB   25            17
4     5    Austin Ekeler  LAC       RB   26            16

   Passing Completions  Passing Attempts  Passsing Yards  Passing TDs  …  \
0                    0                 0               0            0  …
1                    0                 1               0            0  …
2                    1                 2              24            1  …
3                  409               646            4407           36  …
4                    0                 0               0            0  …

   Rushing TDs  Target  Receptions  Receiving Yards  Yards Per Reception  \
0           18      51          40              360                 9.00
1            0     191         145             1947                13.43
2            8     121          77             1405                18.25
3            6       0           0                0                  NaN
4           12      94          70              647                 9.24

   Receiving Yards.1  Fumbles Lost  Total TDs  Fantasy Points  PPR Points
0                  2             2         20             333       373.1
1                 16             0         16             295       439.5
2                  6             2         14             262       339.0
3                  0             3          6             403       402.6
4                  8             3         20             274       343.8

[5 rows x 24 columns]
2022 Data Preview:
   Rank               Player Team Position  Age  Games Played  \
0     1      Patrick Mahomes  KAN       QB   27            17
1     2          Josh Jacobs  LVR       RB   24            17
2     3  Christian McCaffrey  2TM       RB   26            17
3     4        Derrick Henry  TEN       RB   28            16
4     5     Justin Jefferson  MIN       WR   23            17

   Passing Completions  Passing Attempts  Passing Yards  Passing Touchdowns  \
0                  435               648           5250                  41
1                    0                 0              0                   0
```

```
2                     1                  1            34                    1
3                     2                  2             4                    1
4                     2                  2            34                    0

   …  Rushing TD  Targets  Receptions  Receiving Yards  \
0  …           4        1           1                6
1  …          12       64          53              400
2  …           8      108          85              741
3  …          13       41          33              398
4  …           1      184         128             1809

   Yards per Receptions  Reciving Touchdowns  Fumbles Lost  Total TD2  \
0                  6.00                    0             5          4
1                  7.55                    0             3         12
2                  8.72                    5             1         13
3                 12.06                    0             6         13
4                 14.13                    8             0          9

   Fantasy Points  PPR Fantasy Points
0             416               417.4
1             275               328.3
2             271               356.4
3             270               302.8
4             241               368.7

[5 rows x 24 columns]
```

[5]:

```python
# Check for missing values and data types for each year
print("2019 Data Information:")
print(df_2019.info())
print("\n2020 Data Information:")
print(df_2020.info())
print("\n2021 Data Information:")
print(df_2021.info())
print("\n2022 Data Information:")
print(df_2022.info())
```

```
2019 Data Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Rank                  200 non-null    int64
 1   Player                200 non-null    object
 2   Team                  200 non-null    object
 3   Position              200 non-null    object
 4   Age                   200 non-null    int64
```

```
 5   Games Played         200 non-null    int64
 6   Passing Completion   200 non-null    int64
 7   Passing Attempts     200 non-null    int64
 8   Passing Yards        200 non-null    int64
 9   Passing TDs          200 non-null    int64
 10  Interceptions        200 non-null    int64
 11  Rushing Attempts     200 non-null    int64
 12  Rushing Yards        200 non-null    int64
 13  Yards per Attempt    156 non-null    float64
 14  Rushing TDs          200 non-null    int64
 15  Targets              200 non-null    int64
 16  Recepotions          200 non-null    int64
 17  Receiving Yards      200 non-null    int64
 18  Yards per Reception  167 non-null    float64
 19  Receiving TDs        200 non-null    int64
 20  Fumbles Lost         200 non-null    int64
 21  Total TD             200 non-null    int64
 22  Fantasy Points       200 non-null    int64
 23  PPR Fantasy Points   200 non-null    float64
dtypes: float64(3), int64(18), object(3)
memory usage: 37.6+ KB
None

2020 Data Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 24 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Rank                 200 non-null    int64
 1   Player               200 non-null    object
 2   Team                 200 non-null    object
 3   Position             200 non-null    object
 4   Age                  200 non-null    int64
 5   Games Played         200 non-null    int64
 6   Passing Completions  200 non-null    int64
 7   Passing Attempts     200 non-null    int64
 8   Passing Yards        200 non-null    int64
 9   Passing TD           200 non-null    int64
 10  Interceptions        200 non-null    int64
 11  Rushing Attempts     200 non-null    int64
 12  Rushing Yards        200 non-null    int64
 13  Yards Per Attempt    155 non-null    float64
 14  Rushing TD           200 non-null    int64
 15  Targets              200 non-null    int64
 16  Receptions           200 non-null    int64
 17  Receiving Yards      200 non-null    int64
 18  Yards Per Reception  172 non-null    float64
```

```
 19  Receiving TD         200 non-null    int64
 20  Fumles Lost          200 non-null    int64
 21  Total TD             200 non-null    int64
 22  Fantasy Points       200 non-null    int64
 23  PPR Points           200 non-null    float64
dtypes: float64(3), int64(18), object(3)
memory usage: 37.6+ KB
None

2021 Data Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 24 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Rank                 200 non-null    int64
 1   Player               200 non-null    object
 2   Team                 200 non-null    object
 3   Position             200 non-null    object
 4   Age                  200 non-null    int64
 5   Games Played         200 non-null    int64
 6   Passing Completions  200 non-null    int64
 7   Passing Attempts     200 non-null    int64
 8   Passsing Yards       200 non-null    int64
 9   Passing TDs          200 non-null    int64
 10  Interceptions        200 non-null    int64
 11  Rushing Attempts     200 non-null    int64
 12  Rushing Yards        200 non-null    int64
 13  Yards Per Attempt    162 non-null    float64
 14  Rushing TDs          200 non-null    int64
 15  Target               200 non-null    int64
 16  Receptions           200 non-null    int64
 17  Receiving Yards      200 non-null    int64
 18  Yards Per Reception  164 non-null    float64
 19  Receiving Yards.1    200 non-null    int64
 20  Fumbles Lost         200 non-null    int64
 21  Total TDs            200 non-null    int64
 22  Fantasy Points       200 non-null    int64
 23  PPR Points           200 non-null    float64
dtypes: float64(3), int64(18), object(3)
memory usage: 37.6+ KB
None

2022 Data Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
```

```
---  ------                 --------------  -----
 0   Rank                   200 non-null    int64
 1   Player                 200 non-null    object
 2   Team                   200 non-null    object
 3   Position               200 non-null    object
 4   Age                    200 non-null    int64
 5   Games Played           200 non-null    int64
 6   Passing Completions    200 non-null    int64
 7   Passing Attempts       200 non-null    int64
 8   Passing Yards          200 non-null    int64
 9   Passing Touchdowns     200 non-null    int64
 10  Interceptions          200 non-null    int64
 11  Rushing Attempts       200 non-null    int64
 12  Rushing Yards          200 non-null    int64
 13  Yards per Attempt      157 non-null    float64
 14  Rushing TD             200 non-null    int64
 15  Targets                200 non-null    int64
 16  Receptions             200 non-null    int64
 17  Receiving Yards        200 non-null    int64
 18  Yards per Receptions   166 non-null    float64
 19  Reciving Touchdowns    200 non-null    int64
 20  Fumbles Lost           200 non-null    int64
 21  Total TD2              200 non-null    int64
 22  Fantasy Points         200 non-null    int64
 23  PPR Fantasy Points     200 non-null    float64
dtypes: float64(3), int64(18), object(3)
memory usage: 37.6+ KB
None
```

[6]:
```
# Basic statistics for numerical columns (mean, min, max, etc.)
print("\n2019 Summary Statistics:")
print(df_2019.describe())
print("\n2020 Summary Statistics:")
print(df_2020.describe())
print("\n2021 Summary Statistics:")
print(df_2021.describe())
print("\n2022 Summary Statistics:")
print(df_2022.describe())
```

```
2019 Summary Statistics:
             Rank         Age  Games Played  Passing Completion  \
count  200.000000  200.000000    200.000000          200.000000
mean   100.500000   26.330000     14.105000           51.680000
std     57.879185    3.846437      2.460538          115.946174
min      1.000000   21.000000      3.000000            0.000000
25%     50.750000   24.000000     13.000000            0.000000
50%    100.500000   25.500000     15.000000            0.000000
```

```
75%    150.250000   28.000000     16.000000            0.000000
max    200.000000   42.000000     17.000000          408.000000

       Passing Attempts  Passing Yards   Passing TDs  Interceptions  \
count        200.000000     200.000000    200.000000     200.000000
mean          80.795000     592.350000      3.775000       1.755000
std          181.237935    1334.654668      8.575082       4.454682
min            0.000000       0.000000      0.000000       0.000000
25%            0.000000       0.000000      0.000000       0.000000
50%            0.000000       0.000000      0.000000       0.000000
75%            1.000000       0.000000      0.000000       0.000000
max          626.000000    5109.000000     36.000000      30.000000

       Rushing Attempts  Rushing Yards  …  Rushing TDs     Targets  \
count        200.000000     200.000000  …   200.000000  200.000000
mean          55.970000     246.340000  …     1.960000   59.705000
std           81.342066     366.605212  …     3.163612   44.292404
min            0.000000     -12.000000  …     0.000000    0.000000
25%            1.000000       0.000000  …     0.000000   21.750000
50%            9.000000      40.500000  …     0.000000   56.500000
75%           82.250000     374.250000  …     3.000000   90.250000
max          303.000000    1540.000000  …    16.000000  185.000000

       Recepotions  Receiving Yards  Yards per Reception  Receiving TDs  \
count   200.000000       200.000000           167.000000     200.000000
mean     40.120000       471.120000            11.116766       3.055000
std      29.785086       388.879547             3.683657       2.760448
min       0.000000        -4.000000            -4.000000       0.000000
25%      14.750000       118.750000             8.325000       0.000000
50%      39.000000       424.500000            11.180000       3.000000
75%      59.000000       715.250000            13.720000       5.000000
max     149.000000      1725.000000            20.690000      11.000000

       Fumbles Lost    Total TD  Fantasy Points  PPR Fantasy Points
count    200.000000  200.000000      200.000000          200.000000
mean       1.090000    5.040000      135.945000          176.032000
std        1.585709    3.275982       70.621448           73.096789
min        0.000000    0.000000       55.000000           58.100000
25%        0.000000    3.000000       79.750000          113.575000
50%        1.000000    5.000000      118.500000          164.550000
75%        2.000000    7.000000      168.000000          225.500000
max       11.000000   19.000000      416.000000          471.200000

[8 rows x 21 columns]

2020 Summary Statistics:
             Rank         Age  Games Played  Passing Completions  \
count  200.000000  200.000000    200.000000            200.00000
```

mean	100.500000	26.345000	14.040000	54.08000
std	57.879185	3.932339	2.598492	118.08405
min	1.000000	21.000000	3.000000	0.00000
25%	50.750000	24.000000	13.000000	0.00000
50%	100.500000	25.000000	15.000000	0.00000
75%	150.250000	28.000000	16.000000	0.25000
max	200.000000	43.000000	16.000000	407.00000

	Passing Attempts	Passing Yards	Passing TD	Interceptions	\
count	200.00000	200.000000	200.000000	200.000000	
mean	82.13500	601.650000	4.085000	1.715000	
std	178.39076	1325.122438	9.709229	3.758244	
min	0.00000	0.000000	0.000000	0.000000	
25%	0.00000	0.000000	0.000000	0.000000	
50%	0.00000	0.000000	0.000000	0.000000	
75%	1.00000	1.000000	0.000000	0.000000	
max	626.00000	4823.000000	48.000000	15.000000	

	Rushing Attempts	Rushing Yards	…	Rushing TD	Targets	\
count	200.000000	200.000000	…	200.000000	200.000000	
mean	54.825000	248.605000	…	2.335000	59.025000	
std	74.243687	351.093982	…	3.442678	44.046508	
min	0.000000	-8.000000	…	0.000000	0.000000	
25%	1.000000	0.000000	…	0.000000	19.000000	
50%	11.500000	47.500000	…	1.000000	59.000000	
75%	97.000000	429.500000	…	3.000000	92.250000	
max	378.000000	2027.000000	…	17.000000	166.000000	

	Receptions	Receiving Yards	Yards Per Reception	Receiving TD	\
count	200.00000	200.000000	172.000000	200.00000	
mean	40.38000	459.670000	10.541744	3.22000	
std	29.97764	385.912234	3.943376	3.32821	
min	0.00000	-6.000000	-6.000000	0.00000	
25%	16.00000	122.250000	7.740000	0.00000	
50%	38.00000	418.000000	10.760000	3.00000	
75%	59.00000	723.750000	13.222500	5.00000	
max	127.00000	1535.000000	20.910000	18.00000	

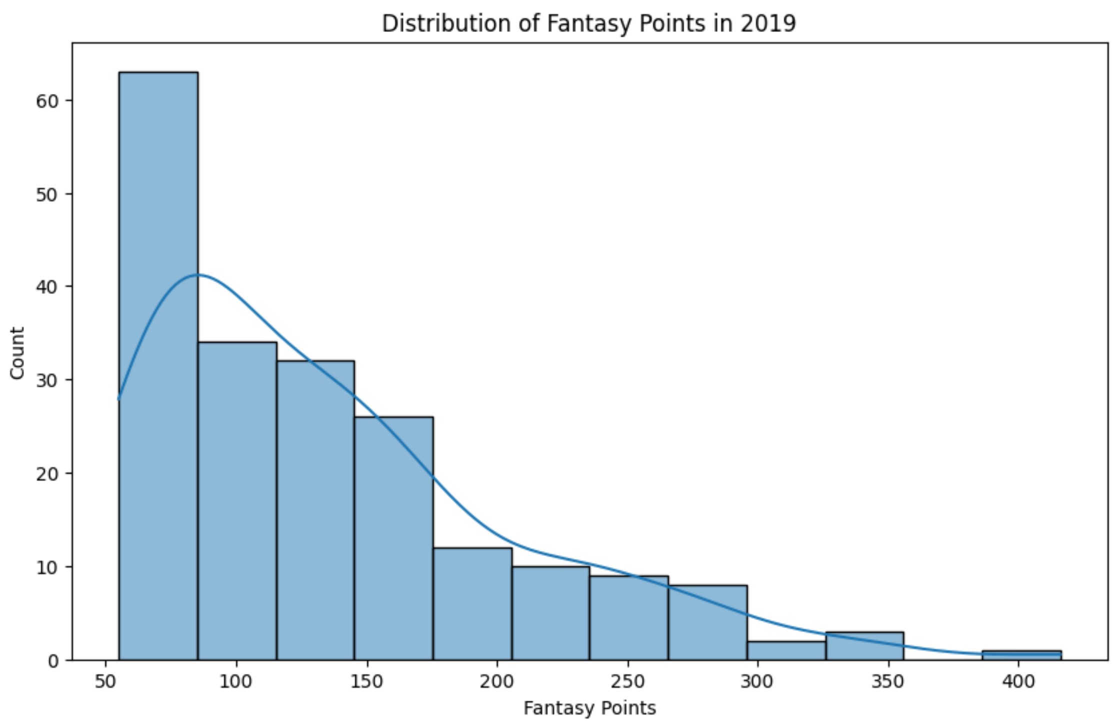
	Fumles Lost	Total TD	Fantasy Points	PPR Points
count	200.00000	200.000000	200.000000	200.000000
mean	0.93000	5.580000	140.285000	180.623500
std	1.39457	3.631742	74.926052	75.091079
min	0.00000	0.000000	63.000000	64.300000
25%	0.00000	3.000000	86.000000	126.875000
50%	0.00000	5.000000	116.000000	164.200000
75%	1.00000	7.000000	166.000000	223.725000
max	8.00000	21.000000	395.000000	396.100000

```
[8 rows x 21 columns]

2021 Summary Statistics:
             Rank         Age  Games Played  Passing Completions  \
count  200.000000  200.000000    200.000000            200.00000
mean   100.500000   26.270000     14.570000             55.75000
std     57.879185    3.663908      2.852329            123.65529
min      1.000000   21.000000      6.000000              0.00000
25%     50.750000   24.000000     13.000000              0.00000
50%    100.500000   26.000000     16.000000              0.00000
75%    150.250000   28.000000     17.000000              0.25000
max    200.000000   44.000000     17.000000            485.00000

       Passing Attempts  Passsing Yards  Passing TDs  Interceptions  \
count        200.000000      200.000000   200.000000     200.000000
mean          85.710000      614.010000     3.935000       1.945000
std          188.009798     1365.233155     9.389975       4.304021
min            0.000000        0.000000     0.000000       0.000000
25%            0.000000        0.000000     0.000000       0.000000
50%            0.000000        0.000000     0.000000       0.000000
75%            1.000000        1.000000     0.000000       0.000000
max          719.000000     5316.000000    43.000000      17.000000

       Rushing Attempts  Rushing Yards  …  Rushing TDs      Target  \
count        200.000000     200.000000  …    200.00000  200.000000
mean          59.350000     263.215000  …      2.19000   60.150000
std           77.035184     342.780954  …      3.20237   46.423737
min            0.000000       0.000000  …      0.00000    0.000000
25%            1.000000       5.000000  …      0.00000   20.000000
50%           17.500000      78.500000  …      1.00000   57.500000
75%          104.250000     435.250000  …      3.00000   92.250000
max          332.000000    1811.000000  …     18.00000  191.000000

       Receptions  Receiving Yards  Yards Per Reception  Receiving Yards.1  \
count   200.00000       200.000000           164.000000         200.000000
mean     41.16500       465.720000            10.696951           3.015000
std      30.97718       401.140274             3.520453           3.224401
min       0.00000        -4.000000            -4.000000           0.000000
25%      18.00000       128.750000             7.977500           0.000000
50%      41.00000       430.500000            10.690000           2.000000
75%      61.00000       705.000000            13.122500           5.000000
max     145.00000      1947.000000            19.540000          16.000000

       Fumbles Lost   Total TDs  Fantasy Points  PPR Points
count    200.000000  200.000000      200.000000  200.000000
mean       0.960000    5.220000      139.775000  180.869000
std        1.306497    3.691849       73.146335   75.794843
min        0.000000    0.000000       59.000000   62.500000
```

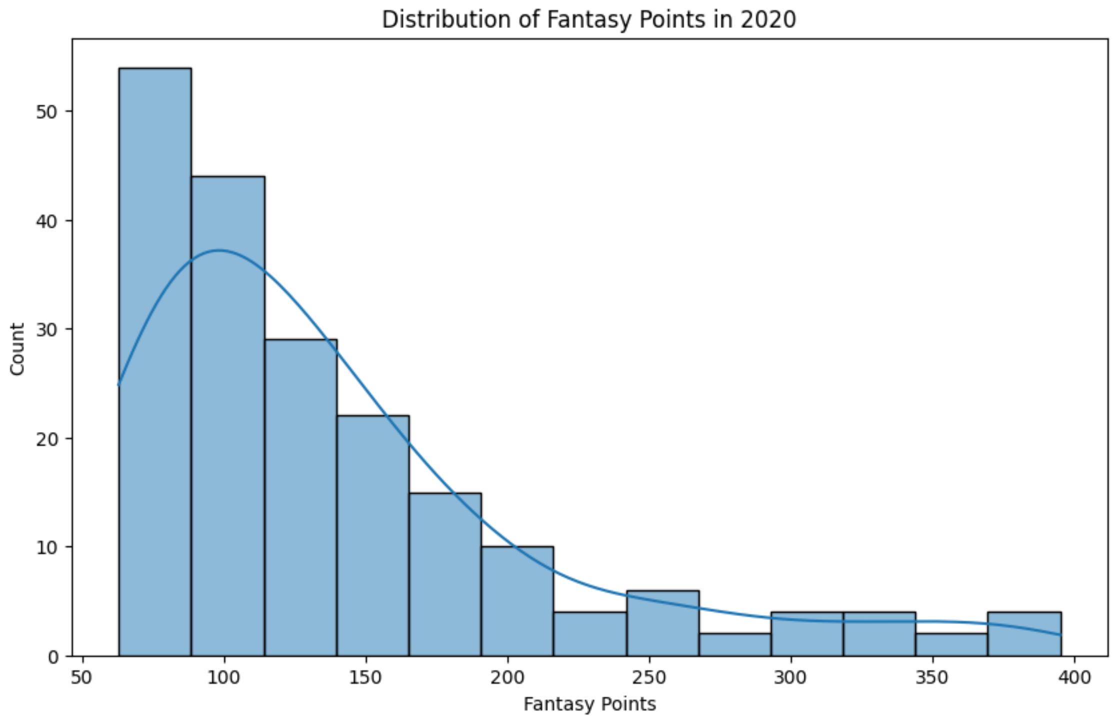
```
25%        0.000000    3.000000       85.000000  121.700000
50%        1.000000    5.000000      116.500000  164.200000
75%        1.000000    7.000000      172.000000  227.350000
max        6.000000   20.000000      403.000000  439.500000

[8 rows x 21 columns]

2022 Summary Statistics:
             Rank         Age  Games Played  Passing Completions  \
count  200.000000  200.000000    200.000000           200.000000
mean   100.500000   26.385000     14.585000            50.680000
std     57.879185    3.438881      2.760594           114.872346
min      1.000000   21.000000      6.000000             0.000000
25%     50.750000   24.000000     13.000000             0.000000
50%    100.500000   26.000000     16.000000             0.000000
75%    150.250000   28.000000     17.000000             0.000000
max    200.000000   45.000000     17.000000           490.000000

       Passing Attempts  Passing Yards  Passing Touchdowns  Interceptions  \
count        200.000000     200.000000          200.000000     200.000000
mean          78.235000     558.965000            3.445000       1.675000
std          175.443014    1256.461792            8.089759       3.744259
min            0.000000       0.000000            0.000000       0.000000
25%            0.000000       0.000000            0.000000       0.000000
50%            0.000000       0.000000            0.000000       0.000000
75%            1.000000       0.000000            0.000000       0.000000
max          733.000000    5250.000000           41.000000      15.000000

       Rushing Attempts  Rushing Yards  …  Rushing TD     Targets  \
count        200.000000     200.000000  …  200.000000  200.000000
mean          60.630000     274.515000  …    2.110000   59.965000
std           83.747845     388.655432  …    3.210833   45.957089
min            0.000000     -15.000000  …    0.000000    0.000000
25%            1.000000       0.000000  …    0.000000   18.000000
50%           10.000000      53.500000  …    1.000000   59.000000
75%           95.000000     462.250000  …    3.000000   92.250000
max          349.000000    1653.000000  …   17.000000  184.000000

       Receptions  Receiving Yards  Yards per Receptions  Reciving Touchdowns  \
count  200.000000        200.00000            166.000000           200.000000
mean    40.730000        456.14500             10.348193             2.765000
std     30.634613        396.47656              3.571122             2.867208
min      0.000000        -10.00000             -5.000000             0.000000
25%     15.750000         95.75000              7.682500             0.000000
50%     40.000000        423.50000             10.585000             2.000000
75%     60.250000        710.75000             12.872500             4.000000
max    128.000000       1809.00000             18.080000            14.000000
```

```
       Fumbles Lost   Total TD2  Fantasy Points  PPR Fantasy Points
count     200.00000  200.000000       200.00000          200.000000
mean        2.11000    4.905000       134.09000          174.766000
std         2.79229    3.319801        71.48962           74.628082
min         0.00000    0.000000        57.00000           56.600000
25%         0.00000    3.000000        79.75000          115.100000
50%         1.00000    4.000000       115.00000          162.600000
75%         3.00000    6.000000       165.75000          219.550000
max        16.00000   18.000000       416.00000          417.400000

[8 rows x 21 columns]
```

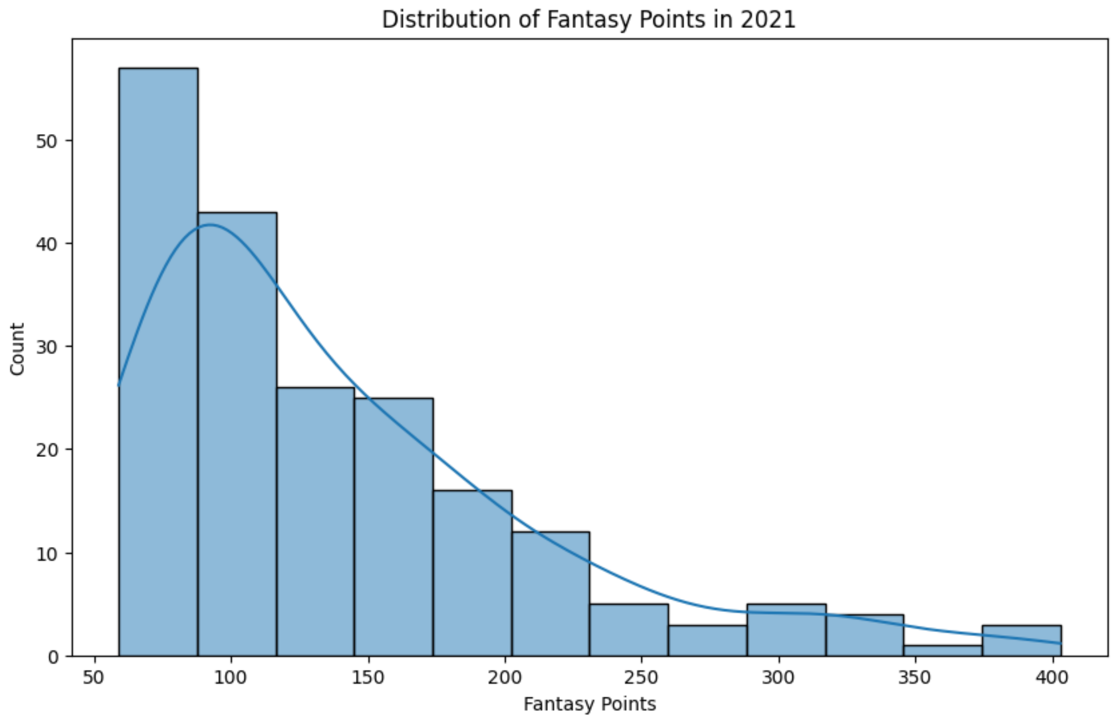
[7]:
```python
# Plot distribution of fantasy points
plt.figure(figsize=(10, 6))
sns.histplot(df_2019['Fantasy Points'], kde=True)
plt.title('Distribution of Fantasy Points in 2019')
plt.show()
```



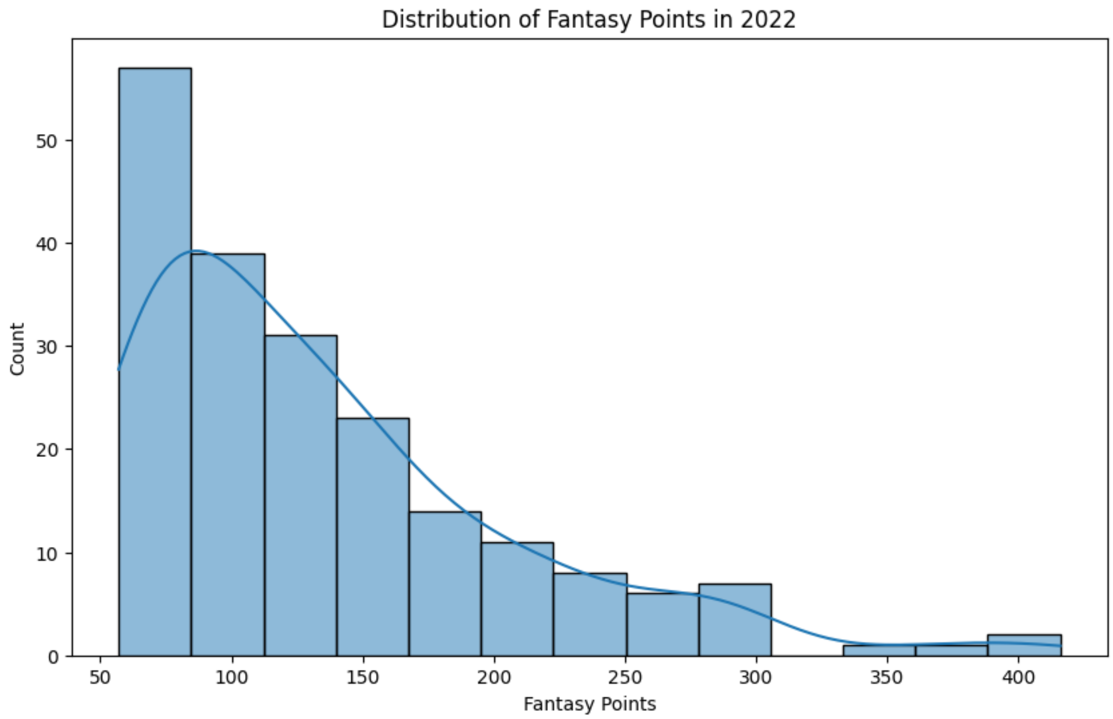
[8]:
```python
# Plot distribution of fantasy points
plt.figure(figsize=(10, 6))
sns.histplot(df_2020['Fantasy Points'], kde=True)
plt.title('Distribution of Fantasy Points in 2020')
plt.show()
```



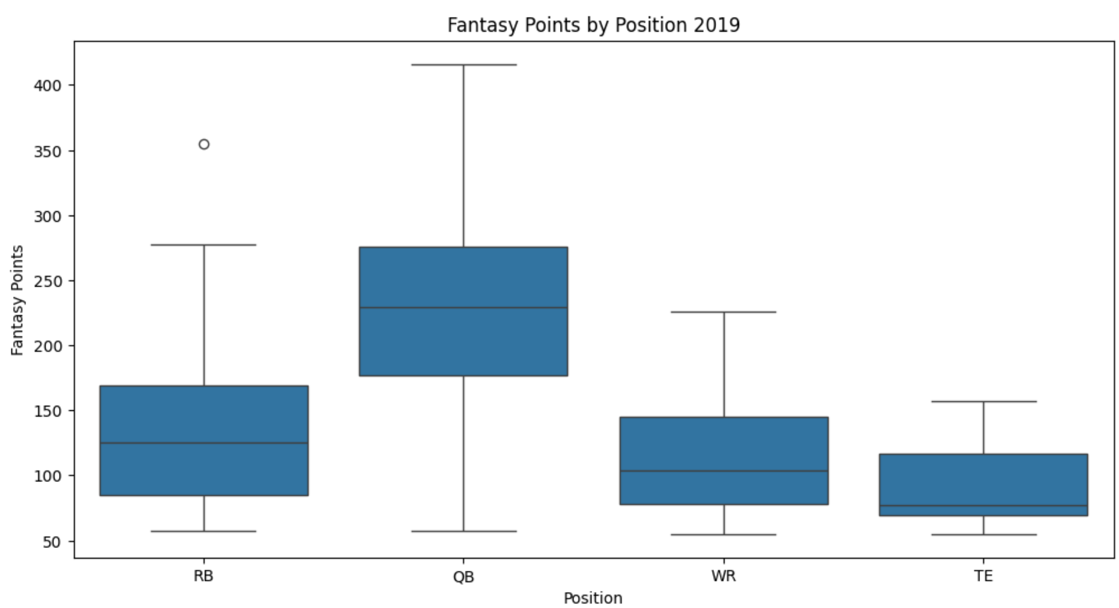
[9]:
```
# Plot distribution of fantasy points
plt.figure(figsize=(10, 6))
sns.histplot(df_2021['Fantasy Points'], kde=True)
plt.title('Distribution of Fantasy Points in 2021')
plt.show()
```



[10]:
```
# Plot distribution of fantasy points
plt.figure(figsize=(10, 6))
sns.histplot(df_2022['Fantasy Points'], kde=True)
plt.title('Distribution of Fantasy Points in 2022')
plt.show()
```

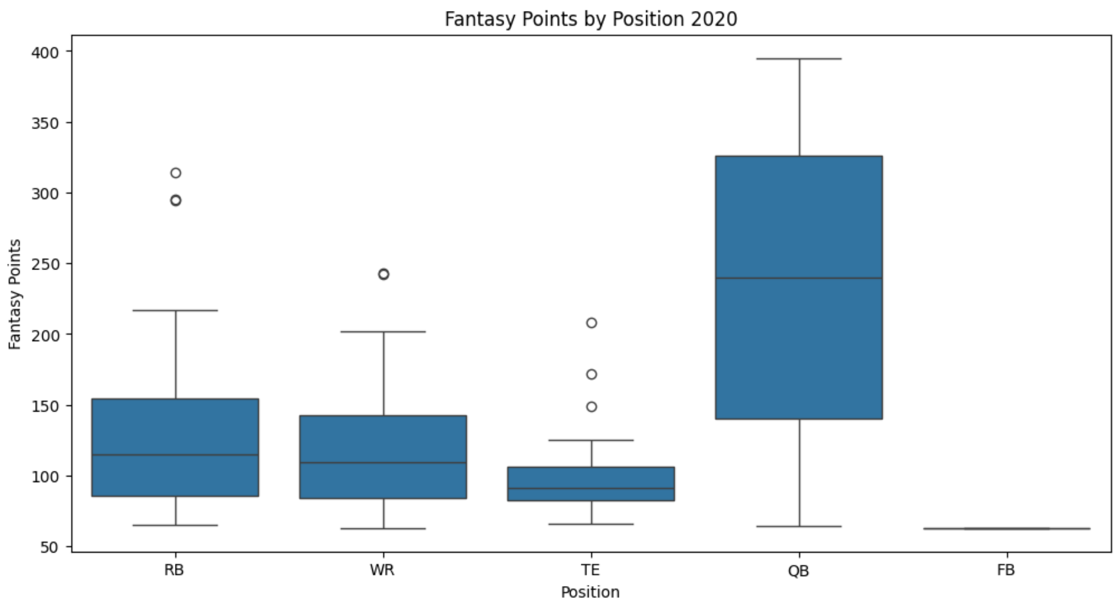


[11]:
```python
# Boxplot to compare fantasy points across positions
plt.figure(figsize=(12, 6))
sns.boxplot(x='Position', y='Fantasy Points', data=df_2019)
plt.title('Fantasy Points by Position 2019')
plt.show()
```



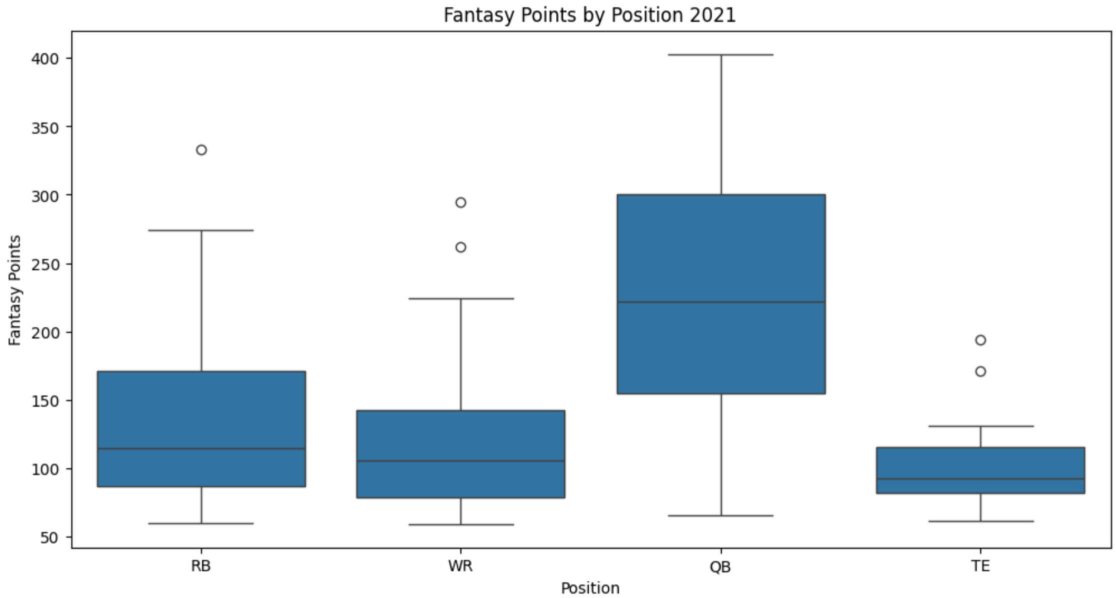
[12]:

```
# Boxplot to compare fantasy points across positions
plt.figure(figsize=(12, 6))
sns.boxplot(x='Position', y='Fantasy Points', data=df_2020)
plt.title('Fantasy Points by Position 2020')
plt.show()
```

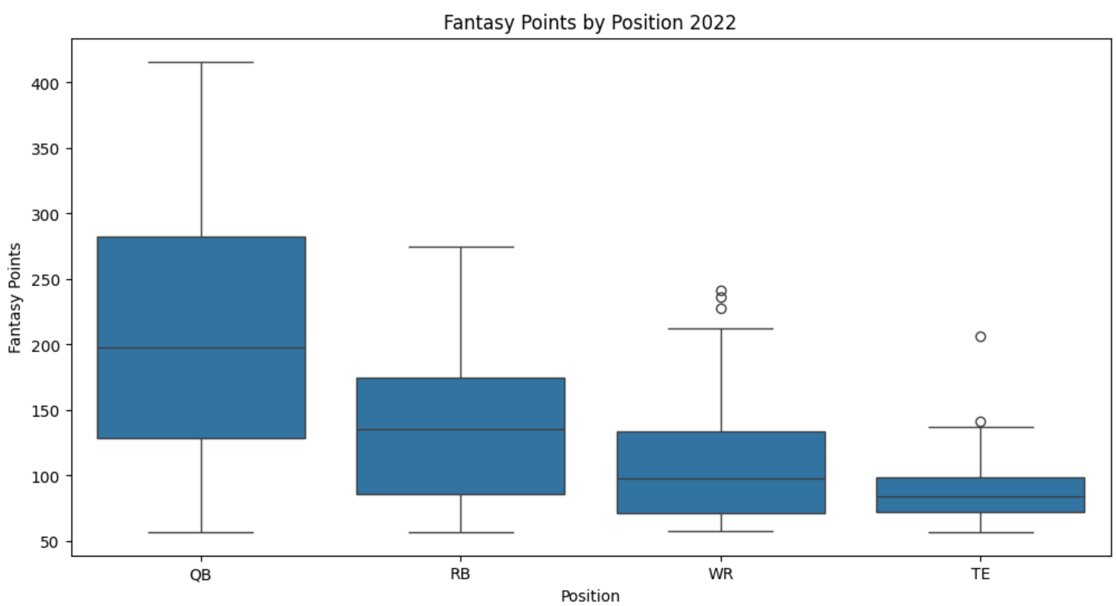


[13]:

```
# Boxplot to compare fantasy points across positions
plt.figure(figsize=(12, 6))
sns.boxplot(x='Position', y='Fantasy Points', data=df_2021)
plt.title('Fantasy Points by Position 2021')
plt.show()
```



[14]:
```python
# Boxplot to compare fantasy points across positions
plt.figure(figsize=(12, 6))
sns.boxplot(x='Position', y='Fantasy Points', data=df_2022)
plt.title('Fantasy Points by Position 2022')
plt.show()
```

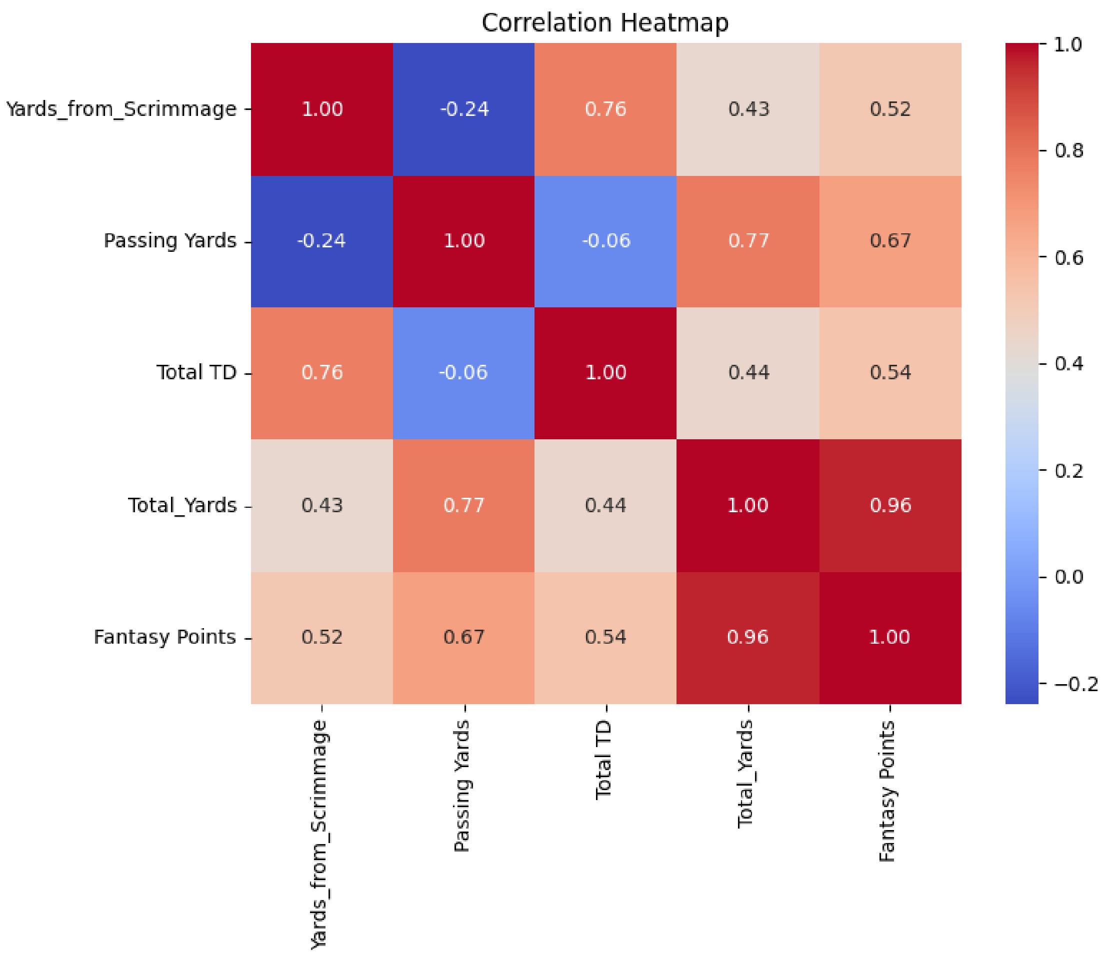


[15]:
```python
# Week 3 Start

# Concatenate all dataframes into a single dataframe
combined_df = pd.concat([df_2019, df_2020, df_2021, df_2022], ignore_index=True)

# Clean player names by removing special characters
combined_df['Player'] = combined_df['Player'].str.replace(r'[^a-zA-Z.\s]', '',␣
↪regex=True)

# Group by 'Player' and aggregate relevant numerical columns
aggregated_df = combined_df.groupby('Player').agg({
    'Rushing Yards': 'sum',
    'Receiving Yards': 'sum',
    'Passing Yards': 'sum',
    'Total TD': 'sum',
    'Fantasy Points': 'sum',
    'Games Played': 'sum',
    'Position': 'first',  # Get the first non-null position
}).reset_index()

# Calculate Yards from Scrimmage and Total Yards
aggregated_df['Yards_from_Scrimmage'] = aggregated_df['Rushing Yards'] +␣
↪aggregated_df['Receiving Yards']
aggregated_df['Total_Yards'] = aggregated_df['Yards_from_Scrimmage'] +␣
↪aggregated_df['Passing Yards']

# Calculate averages for aggregated statistics
aggregated_df['Avg_TD'] = aggregated_df['Total TD'] / aggregated_df['Games␣
↪Played']
aggregated_df['Avg_Yards_from_Scrimmage'] =␣
↪aggregated_df['Yards_from_Scrimmage'] / aggregated_df['Games Played']
aggregated_df['Avg_Passing_Yards'] = aggregated_df['Passing Yards'] /␣
↪aggregated_df['Games Played']
aggregated_df['Avg_Total_Yards'] = aggregated_df['Total_Yards'] /␣
↪aggregated_df['Games Played']

# Save the aggregated data to a new CSV file
aggregated_df.to_csv('aggregated_fantasy_data.csv', index=False)
```

[16]:
```python
# Split Data (Train/Test)
X = aggregated_df[['Yards_from_Scrimmage', 'Passing Yards', 'Total TD',␣
↪'Total_Yards']]
y = aggregated_df['Fantasy Points']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

```
# Confirm the split sizes
print("Train Feature Set Shape:", X_train.shape)
print("Test Feature Set Shape:", X_test.shape)
print("Train Target Set Shape:", y_train.shape)
print("Test Target Set Shape:", y_test.shape)

# Correlation heatmap
plt.figure(figsize=(8, 6))
corr = aggregated_df[['Yards_from_Scrimmage', 'Passing Yards', 'Total TD',␣
↪'Total_Yards', 'Fantasy Points']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```

```
Train Feature Set Shape: (288, 4)
Test Feature Set Shape: (72, 4)
Train Target Set Shape: (288,)
Test Target Set Shape: (72,)
```



[17]:
```
#### Start of Week 4 ####
# Data Cleaning: Remove special characters from 'Player' column
aggregated_df['Player'] = aggregated_df['Player'].str.replace(r'[^a-zA-Z.\s]',␣
↪'', regex=True)
```

[18]:
```
# Handle missing data
imputer = SimpleImputer(strategy='mean')  # You can change the strategy based␣
↪on needs (mean, median, etc.)
aggregated_df['Fantasy Points'] = imputer.fit_transform(aggregated_df[['Fantasy␣
↪Points']])
```

[19]:
```
# Outlier Treatment: Using IQR to detect and remove outliers in Fantasy Points
Q1 = aggregated_df['Fantasy Points'].quantile(0.25)
Q3 = aggregated_df['Fantasy Points'].quantile(0.75)
IQR = Q3 - Q1
# Filtering out outliers beyond 1.5*IQR
aggregated_df = aggregated_df[~((aggregated_df['Fantasy Points'] < (Q1 - 1.5 *␣
↪IQR)) | (aggregated_df['Fantasy Points'] > (Q3 + 1.5 * IQR)))]
```

[20]:
```
# Normalize and Standardize numerical features
scaler = StandardScaler()
numerical_features = ['Yards_from_Scrimmage', 'Passing Yards', 'Total TD',␣
↪'Fantasy Points']
aggregated_df[numerical_features] = scaler.
↪fit_transform(aggregated_df[numerical_features])

# One-hot Encoding for categorical variables
encoder = OneHotEncoder(sparse_output=False, drop='first')  # Dropping first to␣
↪avoid multicollinearity
encoded_position = encoder.fit_transform(aggregated_df[['Position']])
encoded_df = pd.DataFrame(encoded_position, columns=encoder.
↪get_feature_names_out(['Position']))
aggregated_df = pd.concat([aggregated_df, encoded_df], axis=1)
```

[21]:
```
# Remove unnecessary columns (e.g., 'Team' column if not needed)
aggregated_df.drop(columns=['Team'], inplace=True, errors='ignore')

# Handle duplicates by removing any duplicate rows
aggregated_df.drop_duplicates(inplace=True)
```

[22]:
```
# Text Data Cleaning (if applicable): removing stop words, punctuation,␣
↪lowercasing
# This is included as an example in case you have text data, modify if needed
```

```
aggregated_df['Player'] = aggregated_df['Player'].str.lower().str.
 ↪replace(r'[^\w\s]', '', regex=True).str.strip()

# Aggregating relevant statistics (already done previously, no changes needed␣
 ↪for now)
aggregated_df = aggregated_df.groupby('Player').agg({
    'Rushing Yards': 'sum',
    'Receiving Yards': 'sum',
    'Passing Yards': 'sum',
    'Total TD': 'sum',
    'Fantasy Points': 'sum',
    'Games Played': 'sum',
    'Position': 'first',
}).reset_index()
```

[23]:
```
# Save processed data to a new CSV
aggregated_df.to_csv('processed_fantasy_data.csv', index=False)

print("Data processing complete. Here's the head of the cleaned dataframe:")
print(aggregated_df.head())
```

```
Data processing complete. Here's the head of the cleaned dataframe:
            Player  Rushing Yards  Receiving Yards  Passing Yards  Total TD  \
0      aaron jones         2987.0           1615.0      -0.311467  4.027857
1     adam thielen           23.0           2785.0      -0.311467  2.532938
2  adrian peterson         1502.0            243.0      -0.311467  1.038018
3         aj brown           70.0           4491.0      -0.311467  2.532938
4        aj dillon          803.0            519.0      -0.311467 -0.955208

   Fantasy Points  Games Played Position
0        2.730606          62.0       RB
1        1.126495          55.0       WR
2       -0.126419          31.0       RB
3        1.914313          60.0       WR
4        0.106129          34.0       RB
```

[24]:
```
##### Start of Week  5 ########
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler


# Load your dataset
df = pd.read_csv('aggregated_fantasy_data.csv')
```

```python
# Feature 1: Create a new feature 'Total_Yards' by adding rushing and receiving␣
↪yards
df['Total_Yards'] = df['Rushing Yards'] + df['Receiving Yards']

# Feature 2: Create 'Touchdown_Efficiency' by dividing total touchdowns by␣
↪total yards
df['Touchdown_Efficiency'] = df['Total TD'] / df['Total_Yards']

# Feature 3: Create 'Fantasy Points per Game (FP_per_Game)' by dividing fantasy␣
↪points by games played
df['FP_per_Game'] = df['Fantasy Points'] / df['Games Played']

# Drop NaN or infinite values that may result from division
df['Touchdown_Efficiency'].replace([np.inf, -np.inf], np.nan, inplace=True)
df.dropna(subset=['Touchdown_Efficiency'], inplace=True)

# View the new columns added
df[['Player', 'Rushing Yards', 'Receiving Yards', 'Total_Yards', 'Total TD',␣
↪'Touchdown_Efficiency', 'FP_per_Game']].head()
```

```
/tmp/ipykernel_3833786/2416030159.py:21: 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 original object.


  df['Touchdown_Efficiency'].replace([np.inf, -np.inf], np.nan, inplace=True)
```

[24]:
```
          Player  Rushing Yards  Receiving Yards  Total_Yards  Total TD  \
0     A.J. Brown           70.0             4491       4561.0      21.0
1     A.J. Green            0.0             1371       1371.0       2.0
2      AJ Dillon          803.0              519       1322.0       0.0
3    Aaron Jones         2987.0             1615       4602.0      30.0
4  Aaron Rodgers          433.0              -10        423.0       4.0

   Touchdown_Efficiency  FP_per_Game
0              0.004604    11.216667
1              0.001459     5.218750
2              0.000000     8.588235
3              0.006519    13.629032
4              0.009456    18.938462
```

[30]:
```python
# Select only numeric features (excluding player names or other categorical␣
↪features)
numeric_columns = df.select_dtypes(include=['number']).columns
```

[31]:
```python
# Week 6: PCA

df = pd.read_csv('aggregated_fantasy_data.csv')
df['Total_Yards'] = df['Rushing Yards'] + df['Receiving Yards']
# Avoid division by zero
df['Touchdown_Efficiency'] = np.where(df['Total_Yards'] != 0, df['Total TD'] /␣
↪df['Total_Yards'], 0)
df['FP_per_Game'] = df['Fantasy Points'] / df['Games Played']

# Split data first to prevent leakage
X = df[numeric_columns].drop(columns=['Fantasy Points'])
y = df['Fantasy Points']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Feature Scaling and PCA
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train.select_dtypes(include=['int64',␣
↪'float64']))
X_val_scaled = scaler.transform(X_val.select_dtypes(include=['int64',␣
↪'float64']))

pca = PCA(n_components=5)
X_train_pca = pca.fit_transform(X_train_scaled)
X_val_pca = pca.transform(X_val_scaled)

# Model
rf = RandomForestRegressor(random_state=42)
param_grid = {'n_estimators': [100, 200, 300], 'max_depth': [10, 20, None],␣
↪'min_samples_split': [2, 5, 10]}
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3,␣
↪scoring='r2', n_jobs=-1, verbose=2)
grid_search.fit(X_train_pca, y_train)

# Evaluate the model
y_val_pred = grid_search.best_estimator_.predict(X_val_pca)
val_rmse = np.sqrt(mean_squared_error(y_val, y_val_pred))
val_r2 = r2_score(y_val, y_val_pred)

print(f"Validation RMSE: {val_rmse}")
print(f"Validation R^2: {val_r2}")
```

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
```

```
Validation RMSE: 159.2263800401631
Validation R^2: 0.7018433605794435
```

[32]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Base model
rf_base = RandomForestRegressor(random_state=42)
rf_base.fit(X_train, y_train)

# Predictions
y_train_pred_base = rf_base.predict(X_train)
y_val_pred_base = rf_base.predict(X_val)

# Metrics calculation
train_rmse_base = np.sqrt(mean_squared_error(y_train, y_train_pred_base))
val_rmse_base = np.sqrt(mean_squared_error(y_val, y_val_pred_base))

train_r2_base = r2_score(y_train, y_train_pred_base)
val_r2_base = r2_score(y_val, y_val_pred_base)

print(f"Base Model - Training RMSE: {train_rmse_base}, Validation RMSE:␣
↪{val_rmse_base}")
print(f"Base Model - Training R²: {train_r2_base}, Validation R²:␣
↪{val_r2_base}")
```

```
Base Model - Training RMSE: 18.48719200028916, Validation RMSE:
30.64079352395721
Base Model - Training R²: 0.9950859234251188, Validation R²: 0.9889588462379747
```

[33]:

```
from sklearn.model_selection import GridSearchCV

# Define hyperparameters to tune
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10]
}

# Grid Search
rf_tuned = RandomForestRegressor(random_state=42)
grid_search = GridSearchCV(estimator=rf_tuned, param_grid=param_grid, cv=3,␣
↪scoring='r2', n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)

# Predictions
```

```
y_train_pred_tuned = grid_search.best_estimator_.predict(X_train)
y_val_pred_tuned = grid_search.best_estimator_.predict(X_val)

# Metrics calculation
train_rmse_tuned = np.sqrt(mean_squared_error(y_train, y_train_pred_tuned))
val_rmse_tuned = np.sqrt(mean_squared_error(y_val, y_val_pred_tuned))

train_r2_tuned = r2_score(y_train, y_train_pred_tuned)
val_r2_tuned = r2_score(y_val, y_val_pred_tuned)

print(f"Tuned Model - Training RMSE: {train_rmse_tuned}, Validation RMSE: {val_rmse_tuned}")
print(f"Tuned Model - Training R²: {train_r2_tuned}, Validation R²: {val_r2_tuned}")
```

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
Tuned Model - Training RMSE: 16.94819532653518, Validation RMSE:
30.67567504124581
Tuned Model - Training R²: 0.9958700296141166, Validation R²: 0.9889336934028812
```

[34]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Standardize features before applying PCA
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

# Apply PCA
pca = PCA(n_components=5)
X_train_pca = pca.fit_transform(X_train_scaled)
X_val_pca = pca.transform(X_val_scaled)

# Train Random Forest on PCA-transformed data
rf_pca = RandomForestRegressor(random_state=42)
rf_pca.fit(X_train_pca, y_train)

# Predictions
y_train_pred_pca = rf_pca.predict(X_train_pca)
y_val_pred_pca = rf_pca.predict(X_val_pca)

# Metrics calculation
train_rmse_pca = np.sqrt(mean_squared_error(y_train, y_train_pred_pca))
val_rmse_pca = np.sqrt(mean_squared_error(y_val, y_val_pred_pca))

train_r2_pca = r2_score(y_train, y_train_pred_pca)
val_r2_pca = r2_score(y_val, y_val_pred_pca)
```

```
print(f"PCA Model - Training RMSE: {train_rmse_pca}, Validation RMSE:␣
↪{val_rmse_pca}")
print(f"PCA Model - Training R²: {train_r2_pca}, Validation R²: {val_r2_pca}")
```

```
PCA Model - Training RMSE: 30.30089734988608, Validation RMSE:
159.26648103711102
PCA Model - Training R²: 0.9867988733775771, Validation R²: 0.7016931607936405
```

[ ]:

```

```