



INTRODUCTION TO DATA SCIENCE



FOOTBALL PLAYERS

2015-2024

project overview

*Title: Football Player Data
Cleaning and Analysis
(2015-2024)*

*Objective: Clean and
preprocess football player
data to enable insightful
analysis and visualization.*



*let's read the data
head and tail values*



```
[140]: import pandas as pd
dfmpd.read_csv("football.csv")
df.head()
```

```
[141]:
```

	Teams	Seasons	Players	Matches	Goals	Assists	Seasons Ratings
0	InterMiami	2024	Lionel Messi	15	14	11	8.5
1	InterMiami	2023	Lionel Messi	14	11	5	8.0
2	PSG	2022/2023	Lionel Messi	41	21	28	8.1
3	PSG	2021/2022	Lionel Messi	34	11	14	7.9
4	Barcelona	2020/2021	Lionel Messi	47	38	12	8.4

```
[142]: df.tail()
```

```
[143]:
```

	Teams	Seasons	Players	Matches	Goals	Assists	Seasons Ratings
1211	RealSociedad	2020/2021	Mikel Oyarzabal	43	13	10	7.1
1212	RealSociedad	2019/2020	Mikel Oyarzabal	43	13	13	7.1
1213	RealSociedad	2018/2019	Mikel Oyarzabal	41	14	2	6.7
1214	RealSociedad	2017/2018	Mikel Oyarzabal	43	14	6	7.0
1215	RealSociedad	2016/2017	Mikel Oyarzabal	43	4	8	6.6

```
[144]: df.shape
```

memory usage of the dataset

how to reduce
memory size?

```
[220]: import pandas as pd
df = pd.read_csv("football.csv", usecols=req_cols)
df.info(memory_usage="deep")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1216 entries, 0 to 1215
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Teams                 1216 non-null  object
1   Seasons               1216 non-null  object
2   Players               1216 non-null  object
3   Matches               1216 non-null  int64
4   Goals                 1216 non-null  int64
5   Assists               1216 non-null  int64
6   Seasons Ratings       1216 non-null  float64
dtypes: float64(1), int64(3), object(3)
memory usage: 249.2 KB
```

after reducing the memory size

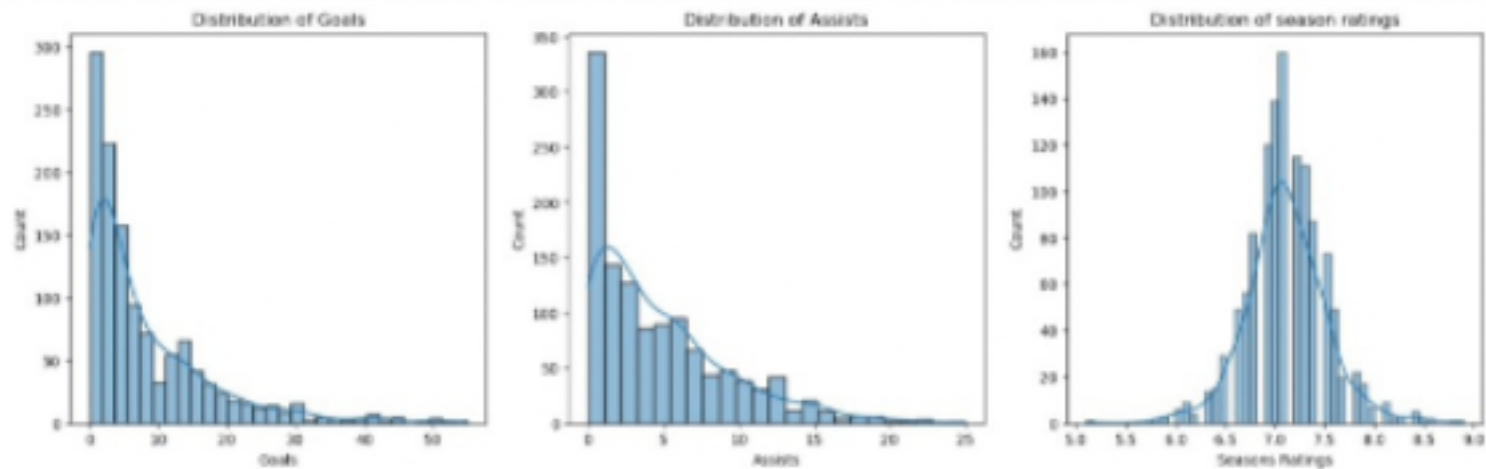
```
[224]: df.info(memory_usage="deep")

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1216 entries, 0 to 1215
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Teams           1216 non-null   object
1   Seasons         1216 non-null   object
2   Players         1216 non-null   object
3   Matches         1216 non-null   int32
4   Goals           1216 non-null   int32
5   Assists         1216 non-null   int32
6   Seasons Ratings 1216 non-null   float32
dtypes: float32(1), int32(3), object(3)
memory usage: 230.2 KB
```

histogram

```
[228]: import matplotlib.pyplot as plt  
import seaborn as sns
```

```
[229]: fig, axs = plt.subplots(1, 3, figsize=(18, 5))  
sns.histplot(df['Goals'], kde=True, ax=axs[0]).set_title('Distribution of Goals')  
  
sns.histplot(df['Assists'], kde=True, ax=axs[1]).set_title('Distribution of Assists')  
sns.histplot(df['Seasons Ratings'], kde=True, ax=axs[2]).set_title('Distribution of season ratings')  
plt.show()
```



accuracy and prediction

```
[11]: dfadf.drop(columns=['Seasons'])

[16]: X = df.drop(columns=['Seasons Ratings'])
      y = df['Seasons Ratings']

[27]: from sklearn.model_selection import train_test_split
      X_train,X_test,y_train,y_test = train_test_split(X,y, random_state=42,test_size=0.2)
      X_train,y_train

[27]: [
      Teams      Players  Matches  Goals  Assists
432    Seville    Jules Kounde    44      3      1
277    Chelsea    Kai Havertz    47     14      5
721    Man City    Manuel Akanji    48      4      0
991  Atletico Madrid  Thomas Partey    24      1      4
678  Atletico Madrid  Rodrigo De Paul    48      4      1
...
1044  Wolfsburg    Victor Osimhen    13      0      0
1095  Juventus    Adrien Rabiot    45      0      2
1198  Brighton    Lewis Dunk    59      1      1
868  Lineres Deportivo  Fernán Lopez    40     12      0
1126  Brighton    Lewis Dunk    31      1      0

[372 rows x 5 columns],
432    7.3
277    7.1
721    7.2
991    6.7
678    7.1
...
1044    5.8
```

```
[282]: y_predict=model.predict(X_train)
```

```
[283]: model.score(X_train,y_train)*100
```

```
[283]: 92.51897106886802
```

accuracy is 92%

prediction

```
[60]: print("predicted values:\n",y_predict)
      print("actual values:\n",y_test)

predicted values:
[7.17400000 7.07899992 7.16299986 6.82599987 7.1159999 6.85000011
 6.67299992 7.07799992 7.37200007 6.77000006 6.77499991 7.01399994
 7.18299985 7.22100008 7.27000005 6.89900012 6.81800007 7.0949999
 7.37000001 7.26900013 6.93800007 7.02299999 6.74199994 7.14816655
 7.18099988 7.48799987 7.43099997 7.29400013 6.80799987 7.22499997
 7.05399987 7.61999988 7.45399998 7.06699998 6.69099993 7.41200008
 7.1049999 7.26400005 7.44099994 6.73099988 6.465 7.45400004
 6.48799997 6.04600015 7.28400001 7.70699988 7.30300007 6.06200005
 6.91399999 6.91700006 7.35700009 7.12900002 6.89700015 7.25300013
 7.47899999 7.03799998 6.897 7.07699994 7.35100013 7.89800005
 7.18199994 6.99599999 6.93500006 7.3530001 7.11099997 7.81500009
 6.99000006 7.14299988 7.18800012 7.44499996 7.07799993 6.72799992
 7.12199986 7.10599994 6.45399987 7.05100008 6.988 8.26100016
 7.22699985 7.20000007 7.022 7.04299999 6.87500012 6.97500004
 7.12599988 6.83399994 7.16699998 7.21599989 7.01699994 6.99200006
 6.63599996 7.07699996 7.05699993 7.38300007 7.09199991 7.44499999
 7.60699998 6.94100013 7.26400001 6.91600005 6.97700002 6.91800013
 7.01800008 6.92500008 7.28100007 6.72799992 6.94399995 7.2150001
 6.59699995 6.66750007 7.19100012 7.289 7.318 6.80199986
 7.16099986 7.95400025 8.46199987 7.14599995 7.12199996 7.17700011
 7.08299994 7.00800005 7.77900024 6.59199992 7.94600001 7.13799988
 7.32200006 7.22199992 6.89300008 6.92300007 7.26300012 6.88500011
 7.22699985 6.72799991 6.7059999 7.1609999 7.2560001 6.98200015
 8.21399975 7.30200001 6.75600012 8.10100029 7.69600006 7.13800001
 6.86600012 7.68100012 7.25700015 7.19199983 7.28400012 6.88000001
 6.99900004 7.20000013 7.50599994 6.99823339 7.15399986 6.97699995
 6.80800002 7.14199995 7.44400009 6.73199996 7.27499995 7.11499992
 7.46900001 6.93600005 7.35800001 7.02499998 7.45999993 6.94300001
 6.972 7.19799988 7.10299994 6.96300006 6.69099995 7.02099998
 8.20400014 6.86900009 7.46100001 7.00900003 7.09999987 6.75099993
 7.09399992 7.37400009 7.26700013 6.43299995 6.91200011 7.35000001
 7.57200012 7.05599996 6.91700001 6.87200006 7.00700001 7.69600008
 7.13299989 6.98500005 7.889 6.75699998 7.05100001 6.90800006
 6.58800008 7.20300014 7.26100012 6.92300006 7.50899987 7.02233333
 7.02499998 7.15700011 6.55999999 6.22899994 7.34300014 7.07700007
 7.27400012 6.305 7.30800014 6.91600006 7.45200004 7.20199986
 6.9100001 7.22200006 8.14599998 7.28400012 7.28899988 7.02899984
 6.6250001 7.04599996 7.06249998 7.20599988 6.93800005 7.02099999
 7.15899992 7.68599989 7.09500001 6.93800005 6.2210001 6.83900015
 6.41100014 6.75999988 7.16000012 6.60199985 6.79999988 7.21599999
 7.98888889 7.01000000 7.00000000 7.00000000 7.00000000 7.00000000]
```

```
7.09499994 6.80400006 7.38900004 7.13499994 7.25899987 7.07800002
5.92899993 6.85049989 7.12699992 7.16599989 7.49499995 7.29100011
7.48800002 7.07500004 6.88300013 7.31100004 7.22399987 7.29100007
6.54300006 7.62699986 7.68900001 7.64199999 8.11899992 6.78700012
6.87100014 7.33900004 7.12299989 6.63099999 7.07199997 7.31500006
7.46400001 6.87600012 7.07199996 6.91899992 7.04299996 7.17799984
6.86700006 7.093 7.51599995 6.91400008 7.07199993 7.392
6.73200011 6.76399991 6.94500002 6.28099994 7.38599999 6.96900002
7.05899993 7.32200012 6.84100011 6.75299988 6.56100005 7.20000012
7.09499993 7.10399996 7.04000007 7.09299986 7.27700012 6.66899997
7.43199994 7.12499985 7.06700006 6.62199997 6.58199999 6.92400015
7.22799991 6.73399995 6.79199985 7.45299995 6.76799994 7.07099991
6.71200001 7.40500001 6.85500011 7.06800014 6.90400008 7.26500009
7.18800014 6.52100006 7.47100006 6.99299997 7.59999997 7.24800011
7.21700013 7.67700006 7.31699997 6.99399999 7.38900002 7.09199992
7.63199997 6.59500002 7.78800004 7.01999999 7.12599991 7.05600002
6.96200004 6.67599993 7.503 7.25300014 7.443 7.33200012
6.99899992 7.07900007 6.97900007 6.76899984 7.34500012 7.30900013
6.838 6.95499998 7.30000005 7.24400011 6.65199999 7.07899993
6.54400004 6.82 7.00900005 6.85799999 7.27100014 7.24500014
7.25800007 7.09799994 7.15899989 6.83100011 7.15299986 7.46100003
6.90400008 7.42600008 7.36600001 7.24000008 6.49999988 7.04000001
6.96300001 7.41199999 6.89400011 7.05399993 7.74200013 7.24800011
7.98499993 6.64299992 7.38 7.27300006 6.99500003 6.89500016
6.95899993 7.009 6.67499995 7.14546653 7.10499989 6.85999987
7.0740001 6.53500006 6.78700014 7.62699992 7.06799992 7.18099989
6.94000008 6.82500014 7.41500001 6.47100014 7.10499991 7.35600005
7.22599989 7.32400005 6.31 7.17999993 7.07599993 7.14699994
7.69100014 6.07300009 7.16399985 6.68399988 6.39599995 6.81000006]
actual values:
541 6.7
259 7.5
43 7.5
1005 7.1
504 7.0
...
420 6.8
243 7.5
59 6.9
1115 6.7
63 6.6
Name: Seasons Ratings, Length: 244, dtype: float32
```

```
[70]: from sklearn.metrics import mean_squared_error
```

time of the dataset

```
[304]: %time
import pandas as pd
df=pd.read_csv("Football.csv")
df
```

CPU times: total: 31.2 ms
wall time: 12.1 ms

```
[304]:
```

	Teams	Seasons	Players	Matches	Goals	Assists	Seasons Ratings
0	InterMiami	2024	Lionel Messi	15	14	11	8.5
1	InterMiami	2023	Lionel Messi	14	11	5	8.0
2	PSG	2022/2023	Lionel Messi	41	21	20	8.1
3	PSG	2021/2022	Lionel Messi	34	11	14	7.9
4	Barcelona	2020/2021	Lionel Messi	47	38	12	8.4
...
1211	Real Sociedad	2020/2021	Mikel Oyarzabal	43	13	10	7.1
1212	Real Sociedad	2019/2020	Mikel Oyarzabal	45	13	13	7.1
1213	Real Sociedad	2018/2019	Mikel Oyarzabal	41	14	2	6.7
1214	Real Sociedad	2017/2018	Mikel Oyarzabal	43	14	6	7.0
1215	Real Sociedad	2016/2017	Mikel Oyarzabal	43	4	8	6.6

after reducing time

```
[88]: %%time
import pandas as pd
df=pd.read_csv("football.csv",usecols=req_cols,iterator=True,chunksize=90)
df
```

```
CPU times: total: 0 ns
Wall time: 5.56 ms
```

outliers

```
[222]: plt.figure(figsize=(10, 8))
ans.scatterplot(data=df, x='Goals', y='Seasons Ratings', hue='Seasons')
plt.title('Goals vs Seasons Ratings')
plt.xlabel('Goals')
plt.ylabel('Seasons Ratings')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper right')
plt.show()
```





THANK YOU



team-04

K.Harshitha(23091A3243)

M.Lakshmid devi(23091A3265)

BT.Bhavesh(23091A3221)

Nandini(FDH)



Zoho Show

ds project.pdf

(This PDF has been generated using Zoho Show)

To create beautiful presentations, download Zoho Show from Play Store <https://zoho.to/cy7>