DSA8002 COURSEWORK 2

NBA – Player of the week

Evan Ganson Saldanha

40246797

esaldanha01@qub.ac.uk

CONTENTS

Contents

QUEEN'S UNIVERSITY BELFAST

1. Introduction

This report gives a detailed explanation of an Analysis on the NBA – Player of the week data frame between the year 1984 and 2000.

Every sport which is played in this world is recorded and its data stored on a large scale. Furthermore, this data is classified and used to analyze the past performances or to predict the future outcomes. In addition, the managers and coaches of the team use these classified data to formulate strategies and to understand their opponents. Predictions were not only made in sports but also in numerous fields such as medicals, stocks, weather and climate, business and many more.

The simple approach to classifying and analyzing the data is using the Python's *pandas* library. *Pandas* being an open-sourced library provides simple to use Data structure, data analysis, and high-performance tools for the Python language. *pandas* help to not switch to a complex domain-specific language like R, instead enables you to complete whole data analysis part using Python.

The NBA – player of the week is a huge dataset with data of all the player winning the title from 1984 to recent times. Analysis such a huge data would require a high-end system and also tons of time. Hence, considering the data collected between the year 1984 and 2000, the data is cleaned, analyzed and utilized using Jupyter tool in which the pandas' library is imported. Jupyter exists to create open-source programming, open norms, and administrations for intelligent figuring crosswise over many programming languages. NumPy is the key library to perform a wide range of computations in Python. Matplotlib is a 2D library used for plotting quality graphs in Python. The bar charts, pie charts, line graphs, histograms, etc., can all be visualized using this library. All these tools and packages aids in the successful analysis of the NBA-Player of the week data and represent them in a readable way.

2. Raw Data

The **source** of the raw data of NBA Player of the week is following the link:

https://www.kaggle.com/jacobbaruch/nba-player-of-the-week

The **purpose** of choosing this data set is to understand the pattern of the NBA players who tend to win the title every week and their team strength.

NBA – Player of the week is a substantial data, with a dimension of (1145, 13) i.e. with 1145 rows and 13 columns. The following are the attributes of the table:

i. ***Age***: player age at the time.
ii. ***Conference***: The area of the team, East / West / Nan
iii. ***Date***: award date
iv. ***Draft Year***: player draft year
v. ***Height***: height of the player in feet-inch
vi. ***Player***: player full name
vii. ***Position***: the original position of the player
viii. ***Season***: full season (start year- end year)
ix. ***Season short***: season-ending year
x. ***Seasons in the league***: seasons in the league up to date
xi. ***Team***: the team of the awarded player
xii. ***Weight***: weight of the player in pounds
xiii. ***Real_value***: If two awards were given at the same week [East & West] the player got 0.5, else 1 point.

The Import of matplotlib, pandas and NumPy libraries is done to analyze the raw data.

```
In [1]: #Importing panda and numpy library
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Reading the raw data from csv file :

```
In [2]: #Displaying the whole data - extraction
nba_raw_data = pd.read_csv("NBA_player_of_the_week.csv")
nba_raw_data.head(15)
```

Out[2]:

	Age	Conference	Date	Draft Year	Height	Player	Position	Season	Season short	Seasons in league	Team	Weight	Real_value
0	29	NaN	Apr 14, 1985	1978	06-05	Micheal Ray Richardson	PG	1984-1985	1985	6	New Jersey Nets	189	1.0
1	23	NaN	Apr 7, 1985	1982	06-06	Derek Smith	SG	1984-1985	1985	2	Los Angeles Clippers	205	1.0
2	28	NaN	Apr 1, 1985	1979	06-06	Calvin Natt	F	1984-1985	1985	5	Denver Nuggets	220	1.0

Fig 2.1: Raw dataset: NBA – player of the week

The data was downloaded from the above URL, and the CSV file is loaded using pd.read_csv function as shown in the above figure 2.1.

3. Sub Setting

Analyzing and classification of such large data needs high-performance systems and takes a valued amount of time. Hence a subset of the main set is chosen and analysis on the necessary data is done under this coursework.

A criterion is taken for choosing them, which is the year with 2000. Fig 3.1 shows the subsetting using criteria to extract the necessary data and storing into a variable *nba_df*.

Choosing the data for years before 2001 :

In [3]:
```
#Displaying a subset of data and storing in nba_df

criteria = (nba_raw_data['Season short'] < 2001)
nba_df = nba_raw_data[criteria]
nba_df.tail()
```

Out[3]:

	Age	Conference	Date	Draft Year	Height	Player	Position	Season	Season short	Seasons in league	Team	Weight	Real_value
355	27	NaN	Dec 5, 1999	1992	07-01	Shaquille O'Neal	C	1999-2000	2000.0	7	Los Angeles Lakers	325	1.0
356	23	NaN	Nov 28, 1999	1997	06-11	Tim Duncan	FC	1999-2000	2000.0	2	San Antonio Spurs	250	1.0
357	23	NaN	Nov 21, 1999	1998	06-06	Vince Carter	SF	1999-2000	2000.0	1	Toronto Raptors	220	1.0
358	23	NaN	Nov 14, 1999	1995	06-11	Kevin Garnett	PF	1999-2000	2000.0	4	Minnesota Timberwolves	240	1.0
359	30	NaN	Nov 7, 1999	1993	06-03	Sam Cassell	G	1999-2000	2000.0	6	Milwaukee Bucks	185	1.0

Fig 3.1: Data extraction: Subset of NBA- Player of the week.

4. Tidy Data

A. Handling the missing values:

Since nba_df still consist the raw data, there is a possibility of having missing data and redundant data.

The code:

```
nba_df.isnull().sum()
```

The above code checks for the null in nba_df and returns the number of missing values in each column.

It gives the following output for the null value check:

```
In [4]: #Checking for missing value
        nba_df.isnull().sum()

Out[4]: Age                   0
        Conference          360
        Date                  0
        Draft Year            0
        Height                0
        Player                0
        Position              0
        Season                0
        Season short          0
        Seasons in league     0
        Team                  0
        Weight                0
        Real_value            2
        dtype: int64
```

The above output clearly shows the existence of Null or the missing values. The first step for tidying the data is the removal of the Null/NaN data. This process can be further grouped as:

i. Eliminate the Column with missing data:

From figure 3.1, we can spot that "Conference" is an entire column with the missing value. As a result, the whole column can be abandoned to proceed with the analysis, as shown in fig. 4.1.i.

```
In [6]: #Removing the column (Conference) with all missing values
        nba_df = nba_df.drop(columns="Conference")

In [7]: #Displayind rows with missing data
        nba_df[nba_df.isnull().any(axis=1)]

Out[7]:
```

	Age	Date	Draft Year	Height	Player	Position	Season	Season short	Seasons in league	Team	Weight	Real_value
18	29	Dec 2, 1984	1977	06-11	Jack Sikma	C	1984-1985	1985.0	7	Seattle SuperSonics	230	NaN
42	32	Nov 24, 1985	1976	06-07	Alex English	SF	1985-1986	1986.0	9	Denver Nuggets	190	NaN

Fig 4.1.i: Eliminating The column "Conference" from nba_df

ii. Forward filling the missing value:

After the removal of the missing value column, the Fig 4.1.i projects the existence of two more missing value in the row with index 18 and 42. Since the values the similar attributes tend to be 1.0, we can Forward fill the previous row value to the missing value column, as shown in Fig 4.1.ii

In [21]:
```
#Fill fowrward method to replace the remaining missing data as they are carry forward values in Season short and real value
nba_df = nba_df.fillna(method = "ffill")
nba_df.iloc[16:20]
```

Out[21]:

	Age	Date	Draft Year	Height	Player	Position	Season	Season short	Seasons in league	Team	Weight	Real_value
16	23	Dec 16, 1984	1982	06-Jun	Derek Smith	SG	1984-1985	1985	2	Los Angeles Clippers	205	1.0
17	25	Dec 9, 1984	1981	06-Sep	Orlando Woolridge	PF	1984-1985	1985	3	Chicago Bulls	215	1.0
18	29	Dec 2, 1984	1977	06-Nov	Jack Sikma	C	1984-1985	1985	7	Seattle SuperSonics	230	1.0
19	28	Nov 25, 1984	1977	06-Jul	Bernard King	SF	1984-1985	1985	7	New York Knicks	205	1.0

In [22]:
```
nba_df.iloc[40:45]
```

Out[22]:

	Age	Date	Draft Year	Height	Player	Position	Season	Season short	Seasons in league	Team	Weight	Real_value
40	27	Dec 8, 1985	1980	06-Oct	Jeff Ruland	C	1985-1986	1986	4	Washington Bullets	240	1.0
41	29	Dec 1, 1985	1978	06-Sep	Larry Bird	SF	1985-1986	1986	6	Boston Celtics	220	1.0
42	32	Nov 24, 1985	1976	06-Jul	Alex English	SF	1985-1986	1986	9	Denver Nuggets	190	1.0
43	23	Nov 17, 1985	1985	07-00	Patrick Ewing	C	1985-1986	1986	0	New York Knicks	240	1.0
44	25	Nov 10, 1985	1981	06-Aug	Buck Williams	PF	1985-1986	1986	4	New Jersey Nets	215	1.0

Fig 4.1.ii: Fill forward to row 18 and 42 under Real value.

Therefore ,

In [10]:
```
#Re-Checking for missing value
nba_df.isnull().sum()
```

Out[10]:
```
Age                  0
Date                 0
Draft Year           0
Height               0
Player               0
Position             0
Season               0
Season short         0
Seasons in league    0
Team                 0
Weight               0
Real_value           0
dtype: int64
```

Table 2: Checking for missing values

B. Normalization – 1NF:

A data frame is said to be in 1st Normal Form (1NF) if it satisfies these conditions:

- the data set contains only atomic values.
- There are no redundant rows.

To convert the data frame to 1NF, the data frame nba_df should undergo the following operations:

- **Converting height from ft to cms**: For types of nba_df, we get the below response:

```
nba_df.dtypes

Age                    int64
Date                  object
Draft Year             int64
Height                object
Player                object
Position              object
Season                object
Season short           int64
Seasons in league      int64
Team                  object
Weight                object
Real_value           float64
dtype: object
```

Table 3: nba_df.dtypes

The data types for the columns Height, weights season short do not match. But first, the height fields should be converted to proper format as they tend to show non-atomic values. Hence converting the height to cms.

```
In [12]: #Converting height from ft to cms
new_ht = []
for ft in nba_df['Height']:
    h_ft = int(ft[0:2])
    h_inch = int(ft[3:5])
    #print(h_ft+" "+h_inch)
    h_inch += h_ft * 12
    h_cm = round(float(h_inch) * 2.54, 2)
    new_ht.append(h_cm)
nba_df['Height'] = pd.Series(new_ht, index=nba_df.index)
nba_df.head(15)
```

Out[12]:

	Age	Date	Draft Year	Height	Player	Position
0	29	Apr 14, 1985	1978	195.58	Micheal Ray Richardson	PG
1	23	Apr 7, 1985	1982	198.12	Derek Smith	SG
2	28	Apr 1, 1985	1979	198.12	Calvin Natt	F
3	37	Mar 24, 1985	1969	218.44	Kareem Abdul-Jabbar	C
4	28	Mar 17, 1985	1978	205.74	Larry Bird	SF

Fig 4.2.i: Conversion of Height from ft-inches to centimeters

- **Converting the data types to proper formats:**
  Once the data is set to appropriate units, the data types can be changed accordingly. Since the issue lied with the miss match of the data types of Height, weight and season short, following is the method to do so:

```
In [13]: #converting the data types to proper formats for season_short to int64, weight to int64
         nba_df = nba_df.astype({"Season short": np.int64,"Weight": np.float64,"Height": np.float64})
         nba_df.dtypes

Out[13]: Age                    int64
         Date                  object
         Draft Year             int64
         Height               float64
         Player                object
         Position              object
         Season                object
         Season short           int64
         Seasons in league      int64
         Team                  object
         Weight               float64
         Real_value           float64
         dtype: object
```

Fig 4.2.ii: Change of data types

- **Renaming the column names to a standard format:**
  Since all the Column names are not in a single standard format, here the names are standardized.

```
#Renaming the column names to standard format
nba_df = nba_df.rename(columns={'Season short':'Season_end','Seasons in league':'Seasons_in_league','Draft Year':'Draft_year'})
nba_df.head(15)
```

	Age	Date	Draft_year	Height	Player	Position	Season	Season_end	Seasons_in_league	Team	Weight	Real_value
0	29	Apr 14, 1985	1978	195.58	Micheal Ray Richardson	PG	1984-1985	1985	6	New Jersey Nets	189.0	1.0
1	23	Apr 7, 1985	1982	198.12	Derek Smith	SG	1984-1985	1985	2	Los Angeles Clippers	205.0	1.0

Fig 4.2.iii: Standardizing the Column name

- **Removing duplicate column:**
  Since the fields are atomic, season column has multiple values, so separating the start and end date, and deleting the end date column, as the value matches with season short.

```
#removing duplicate column. Since season_end and season_short are one and the same,
#removing the second year range from season column
nba_df['Season_start'] = nba_df['Season'].str.extract('(^[0-9]{4})', expand=True).astype(int)
#Removing the column (Season) with all inappropriate values
nba_df = nba_df.drop(columns="Season")
nba_df.head(15)
```

	Age	Date	Draft_year	Height	Player	Position	Season_end	Seasons_in_league	Team	Weight	Real_value	Season_start
0	29	Apr 14, 1985	1978	195.58	Micheal Ray Richardson	PG	1985	6	New Jersey Nets	189.0	1.0	1984
1	23	Apr 7, 1985	1982	198.12	Derek Smith	SG	1985	2	Los Angeles Clippers	205.0	1.0	1984
2	28	Apr 1, 1985	1979	198.12	Calvin Natt	F	1985	5	Denver Nuggets	220.0	1.0	1984
3	37	Mar 24, 1985	1969	218.44	Kareem Abdul-Jabbar	C	1985	15	Los Angeles Lakers	225.0	1.0	1984
4	28	Mar 17, 1985	1978	205.74	Larry Bird	SF	1985	5	Boston Celtics	220.0	1.0	1984

Fig 4.2.iv: Atomic values in Season column

- **Changing the date format:**
  The date is the primary key in this table. But the date cannot be used as sorting parameter as the format is improper and will result in place wrong sorting order (For eg: after sorting the order might be 1,11,7 as the first digit is considered). Hence the date format is changed to yyyy-mm-dd and then sorting using date column.

```
#Changing the date format from mmm-dd-yyyy to yyyy-mm-dd and sorting based on dates
nba_df['Date'] = pd.to_datetime(nba_df['Date'])
nba_df = nba_df.sort_values(by='Date')
nba_df.head(15)
```

	Age	Date	Draft_year	Height	Player	Position	Season_end	Seasons_in_league
22	26	1984-11-04	1981	208.28	Larry Nance	PF	1985	3
21	31	1984-11-11	1976	200.66	Alex English	SF	1985	8
20	29	1984-11-18	1976	208.28	Moses Malone	C	1985	8

Fig 4.2.v: Sorting after changing the date format

- **Assigning unique ID and re-ordering the table columns:**
  Since there is no proper unique ID, a primary key is created and the tables are sorted to get a well organized and classified data frame, which is ready for operations.

```
# Assigning unique ID and re-ordering the table columns
nba_df['Unique_id'] = nba_df.groupby(['Date','Player']).ngroup()
columnsTitles = ['Unique_id','Player', 'Age', 'Height', 'Weight',
                 'Position', 'Team', 'Draft_year','Date', 'Season_start',
                 'Season_end', 'Seasons_in_league', 'Real_value']
nba_df = nba_df.reindex(columns=columnsTitles)
nba_df.head(15)
```

	Unique_id	Player	Age	Height	Weight	Position	Team	Draft_year	Date	Season_start	Season_end	Seasons_in_league	Real_value
22	0	Larry Nance	26	208.28	205.0	PF	Phoenix Suns	1981	1984-11-04	1984	1985	3	1.0
21	1	Alex English	31	200.66	190.0	SF	Denver Nuggets	1976	1984-11-11	1984	1985	8	1.0
20	2	Moses Malone	29	208.28	215.0	C	Philadelphia Sixers	1976	1984-11-18	1984	1985	8	1.0
19	3	Bernard King	28	200.66	205.0	SF	New York Knicks	1977	1984-11-25	1984	1985	7	1.0
18	4	Jack Sikma	29	210.82	230.0	C	Seattle SuperSonics	1977	1984-12-02	1984	1985	7	1.0
17	5	Orlando Woolridge	25	205.74	215.0	PF	Chicago Bulls	1981	1984-12-09	1984	1985	3	1.0
16	6	Derek Smith	23	198.12	205.0	SG	Los Angeles	1982	1984-	1984	1985	2	1.0

Fig 4.2.vi: Unique Id creation and rearranging

5. Simple Statistics

Now that the data frame is ready, basic operations can be applied to get answers for some queries. This is possible by using the numpy library which is imported initially.

1. **Mean**:
   Query: What is the average weight of players getting the Player of the week award?
   Solution:

```
#1. Average weight of players getting the Player of the week award
average_age = np.mean(nba_df['Weight'])
print("Average weight : ",np.round(average_age,3))
```

```
Average weight :  225.939
```

2. **Min and Max:**
   Query: What is the age range of the player who were awarded?
   Solution:

```
#2. range of height
max_height = np.max(nba_df['Height'])
min_height = np.min(nba_df['Height'])
print("Range of height is (",min_height,",",max_height,") cms")
```

```
Range of height is ( 177.8 , 223.52 ) cms
```

3. **Median and percentile:**
   Query: Show the first quartile, median and third quartile for players age
   Solution:

```
#3. Quartiles
print("25th percentile:    ", np.percentile(nba_df['Age'], 25))
print("Median:             ", np.median(nba_df['Age']))
print("75th percentile:    ", np.percentile(nba_df['Age'], 75))
```

```
25th percentile:    25.0
Median:             27.0
75th percentile:    30.0
```

   The describe function can be used to display all the quartile details in one go, as follows:

```
#Using the describe function
nba_df['Weight'].describe()
```

```
count     360.000000
mean      225.938889
std        33.478329
min       150.000000
25%       205.000000
50%       222.500000
75%       250.000000
max       325.000000
Name: Weight, dtype: float64
```

6. Grouping

For grouping, the group-by function is used in python in collaboration with the selected data frame. The group-by can be followed an aggregate function such as count, sum, etc. The reset_index is used to reset the index and change the title of the new column.

Query 1: What are the number of players from each team who got player of the week, between 1984 and 2000?

Solution: (Only displays due to head())

```
nba_df.groupby(['Team']).size().reset_index(name='counts').head()
```

	Team	counts
0	Atlanta Hawks	15
1	Boston Celtics	10
2	Charlotte Hornets	8
3	Chicago Bulls	29
4	Cleveland Cavaliers	8

Query 2: Do player with more experience tend to win the 'Player of the week' title often?

Solution:

```
nba_df.groupby(['Age'])['Player'].count().reset_index(name='counts')
```

	Age	counts
0	20	2
1	21	5
2	22	14
3	23	32
4	24	34
5	25	34
6	26	40
7	27	36
8	28	41

	Age	counts
9	29	28
10	30	33
11	31	21
12	32	13
13	33	11
14	34	7
15	35	5
16	36	2
17	37	2

From the above output, players between the age of 23 to 31 often win the title. Thus, we can inference that both a few years' experiences and young age matters for winning basketball awards.

Query 2: What is the count of various players getting the 'Player of the week' award from each team?

Solution:

```
nba_df.groupby(['Team','Player'])['Unique_id'].count().reset_index(name='counts').head(15)
```

	Team	Player	counts
0	Atlanta Hawks	Dikembe Mutombo	4
1	Atlanta Hawks	Dominique Wilkins	8
2	Atlanta Hawks	Kevin Willis	2
3	Atlanta Hawks	Steve Smith	1
4	Boston Celtics	Larry Bird	7
5	Boston Celtics	Reggie Lewis	1
6	Boston Celtics	Robert Parish	2
7	Charlotte Hornets	Alonzo Mourning	1
8	Charlotte Hornets	Glen Rice	3
9	Charlotte Hornets	Larry Johnson	4
10	Chicago Bulls	Michael Jordan	23
11	Chicago Bulls	Orlando Woolridge	1
12	Chicago Bulls	Scottie Pippen	5
13	Cleveland Cavaliers	Brad Daugherty	1
14	Cleveland Cavaliers	Hot Rod Williams	1

The above result portraits that, Michael Jordon of Chicago Bulls has been awarded the most between the years 1984 and 2000.

7. Visualization

Visualization is any strategy for making images, graphs or animations to impart a message. Through Visualization, it has been a viable method to convey both unique and solid thoughts since the beginning of mankind. The data in huge tables can be simply easily understood by using visualization techniques. Since we use matplotlib to do these operations in Jupyter, here are some Visualization methods to analyze the nba_df data frame:

1. Pie Chart :

Round graph, which is partitioned into slices to outline the numerical frequencies.

The rate of players winning the title every year between 1984 and 2000 :

```
result = nba_df.groupby(['Season_end'])['Unique_id'].count().reset_index(name='counts')
result
x = result['Season_end']
y = result['counts']

plt.rcParams['figure.figsize'] = [15,6]
plt.pie(y, labels=x, autopct='%1.1f%%',shadow=True, startangle=90)
plt.title('Rate of winning players')
plt.legend(title="Legend",loc="upper right")
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

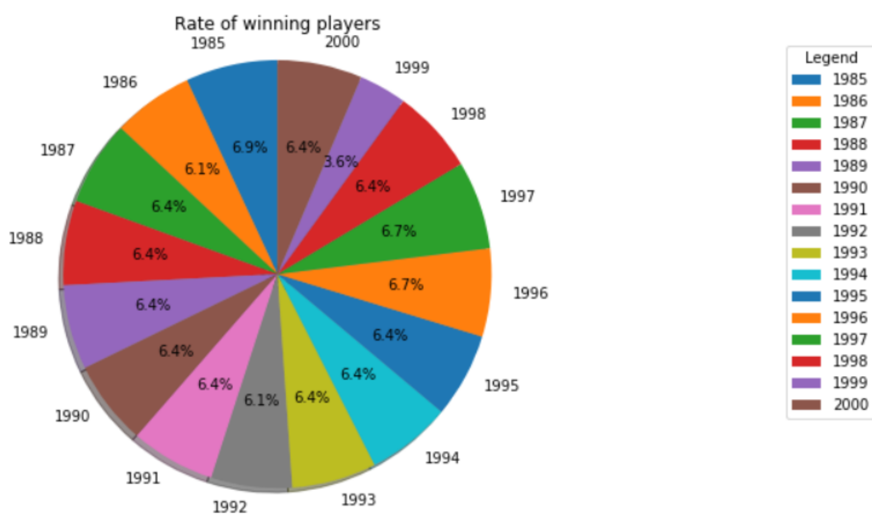

Fig 7.1: Pie chart

The above pie chat represents the winning rate of players every year. We can deduce that the highest number of player awarded was in the year 1985 and the least was 1999.

2. Line Graph / Plot :

A line graph is a two-dimensional representation of a value over a certain time instance. In this project, the line graph is used to solve the query 2 under the Grouping topic:

Query: Do player with more experience tend to win the 'Player of the week' title often?

```
result2 = nba_df.groupby(['Age'])['Player'].count().reset_index(name='counts')
x = result2['Age']
y = result2['counts']

plt.rcParams['figure.figsize'] = [15,6]
plt.plot(x,y, marker = 'o')
plt.grid()
plt.xlabel('Age (Years)')
plt.ylabel('No. of Players')
plt.title("Age vs Player's awarded")
plt.show()
```

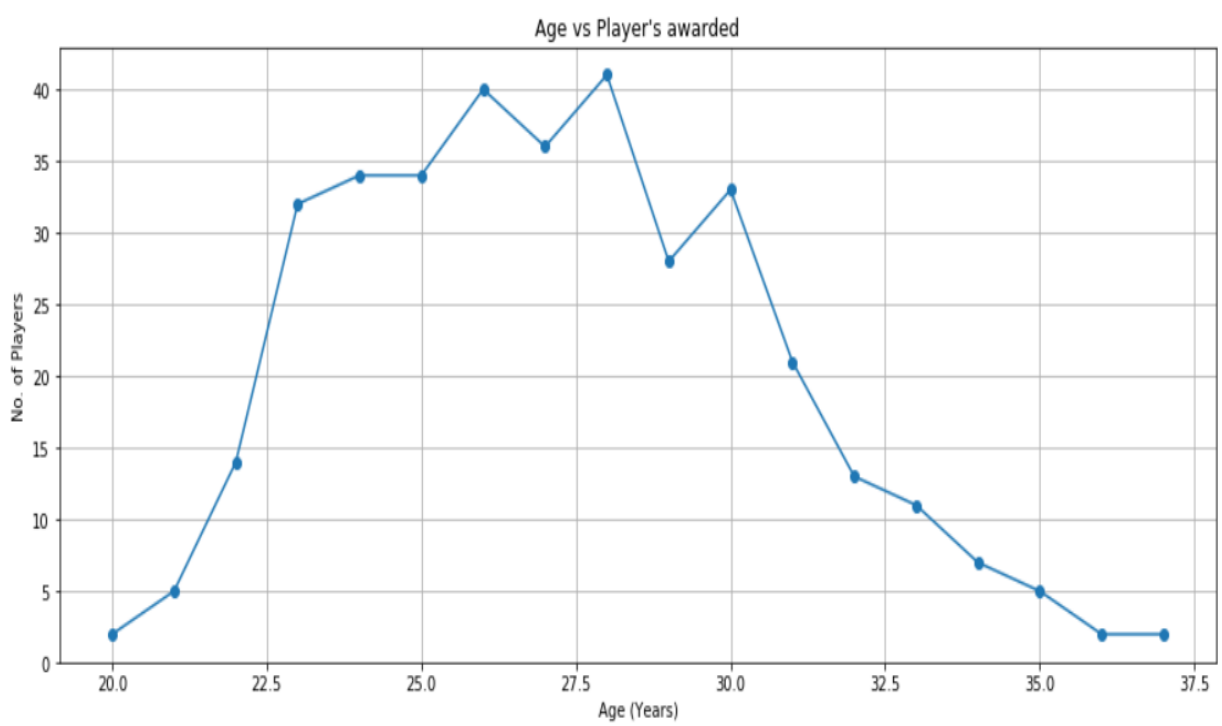

Fig 7.2: Line graph or Line chart

The Line chart clearly indicates that the players aged over 22 and below 31, who are generally experienced and also are young receive the award more often.

3. Bar Graph :

Bar Graphs or the bar plots, as the name suggests, displays information with rectangular bars with lengths or heights corresponding to the values that they resemble.

To see how many players from each team have won the title of player of the week, we can do the following:

```
result3 = nba_df.groupby(['Team']).size().reset_index(name='count')
x = result3['Team']
y = result3['count']

plt.rcParams['figure.figsize'] = [15,6]
plt.bar(x,y)
plt.xticks(rotation = 90)
plt.title("Awards per team")
plt.show()
```

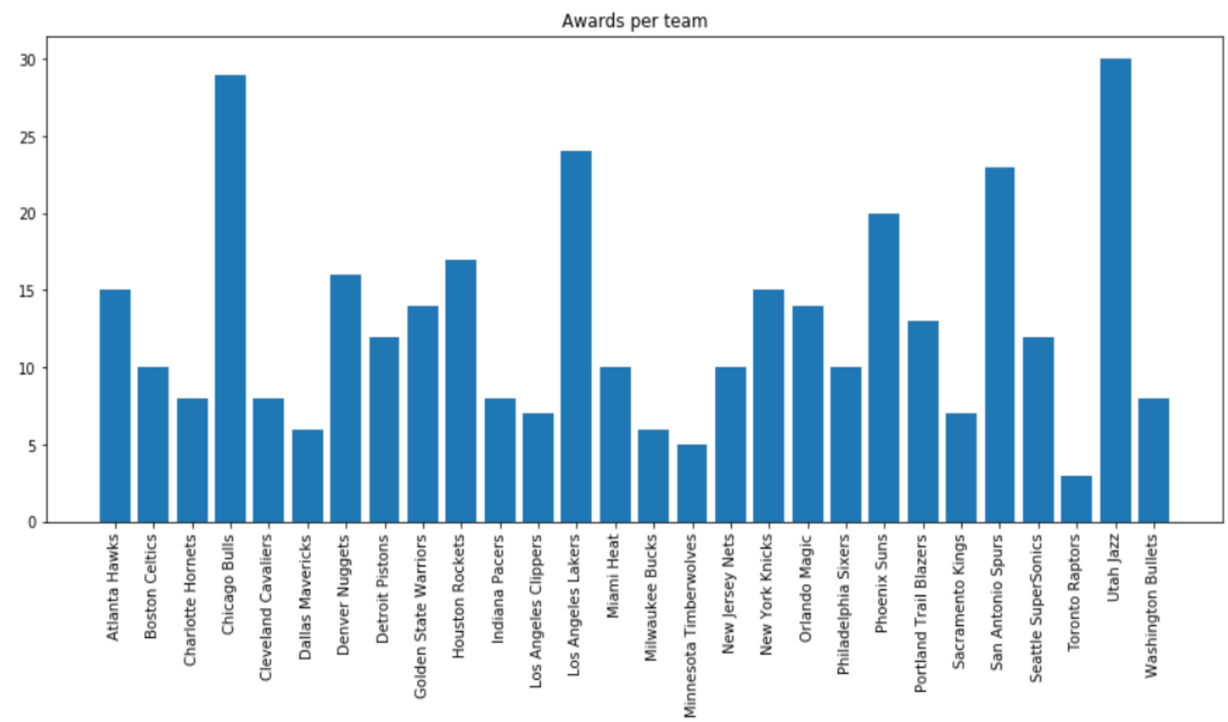

Fig 7.3: Bar Graph

The bar chat explains that Utah Jazz followed by Chicago Bulls have the most number of awarded player in the team. Hence we can inference that these two teams are a force to be reckoned with.

8. Conclusion

Data interpretation plays a key role in every field in this advanced world and every day gives birth to a large amount of data. It is the role of a data scientist to make sense out of this messed up data, which can only be done by using the appropriate data analysis technique. When these techniques are applied, learning from the past and predicting the future won't be a mystery. Such data can be used to predict the exact weather, disaster or accidents and lives can be saved. Therefore, data analysis is a technique that aids every individual to live a life wisely.

9. References

1. Jacob (2018) *NBA player of the week,* Available at *https://www.kaggle.com/jacobbaruch/nba-player-of-the-week* (Accessed: 12 Nov 2018).
2. Michael Heydt (2017) *Learning pandas*, Second edition: Packt Publishing Limited.
3. Ted Petrou (2018) *Selecting Subsets of Data in Pandas,* Available at *https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-39e811c81a0c*(Accessed: 12 Nov 2018).
4. Sejal Jaiswal (2018) *Python Data Type Conversion Tutorial,* Available at *https://www.datacamp.com/community/tutorials/python-data-type-conversion* (Accessed: 13 Nov 2018).
5. The Matplotlib development team (2018) *MatPlotLib: Pie, Bar charts,* Available at *https://matplotlib.org/index.html* (Accessed: 13 Nov 2018).
6. Dr. Yang Hua (Nov 2018) *Lecture Notes and Lab Tutorials*, Queen's Online Resources: Queen's University, Belfast.