CHAPTER I: INTRODUCTION

1.1 PREDICTIVE ANALYTICS

Predictive analytics is a branch of advanced analytics that makes predictions about future outcomes using historical data combined with statistical modeling, data mining techniques and machine learning. Companies employ predictive analytics to find patterns in this data to identify risks and opportunities. Predictive analytics is often associated with big data and data science. Companies today are swimming in data that resides across transactional databases, equipment log files, images, video, sensors or other data sources. To gain insights from this data, data scientists use deep learning and machine learning algorithms to find patterns and make predictions about future events. These include linear and nonlinear regression, neural networks, support vector machines and decision trees. Learnings obtained through predictive analytics can then be used further within prescriptive analytics to drive actions based on predictive insights.

1.2 OVERIVEW OF THE PROJECT

This project is developed for predicting the salary of each players in the IPL 2013 dataset. A dataset is created by IIM (Indian Institute of Management) Bangalore. The Indian Premier League (IPL), sports league was contested during the month of April and May on every year by the teams representing the Indian cities. For predicting the salary of the players we are using TREE BASED algorithms (bagging, boosting, random forest). In our IPL 2013 dataset contains 130 observations and 26 variables.

CHAPTER II – GATHERING DATA

2.1 DATA DESCRIPTION.

The data to be analyzed were to predict cricket player salary for the purpose of discovering whether player salary is high or low. The IPL data frame has 130 rows and 26 columns.

The data frame contains the following columns.

Filter

	SI.NO.	PLAYER.NAME	AGE	COUNTRY	TEAM	PLAYING.ROLE	T.RUNS	T.WKTS	ODI.RUNS.S	ODI.SR.B	ODI.WKTS	ODI.SR.BL	CAPTAINCY.EXP	RUNS.S	HS	AVE	SR.B	SIXERS	RUNS.C	WKTS
1	1	Abdulla, YA	2	SA	KXIP	Allrounder	0	0	0	0.00	0	0.0	0	0	0	0.00	0.00	0	307	
2	2	Abdur Razzak	2	BAN	RCB	Bowler	214	18	657	71.41	185	37.6	0	0	0	0.00	0.00	0	29	
3	3	Agarkar, AB	2	IND	KKR	Bowler	571	58	1269	80.62	288	32.9	0	167	39	18.56	121.01	5	1059	
4	4	Ashwin, R	1	IND	CSK	Bowler	284	31	241	84.56	51	36.8	0	58	11	5.80	76.32	0	1125	
5	5	Badrinath, S	2	IND	CSK	Batsman	63	0	79	45.93	0	0.0	0	1317	71	32.93	120.71	28	0	
6	6	Bailey, GJ	2	AUS	CSK	Batsman	0	0	172	72.26	0	0.0	1	63	48	21.00	95.45	0	0	
7	7	Balaji, L	2	IND	CSK+	Bowler	51	27	120	78.94	34	42.5	0	26	15	4.33	72.22	1	1342	
8	8	Bollinger, DE	2	AUS	CSK	Bowler	54	50	50	92.59	62	31.3	0	21	16	21.00	165.88	1	693	
9	9	Botha, J	2	SA	RR	Allrounder	83	17	609	85.77	72	53.0	1	335	67	30.45	114.73	3	610	
10	10	Boucher, MV	2	SA	RCB+	W. Keeper	5515	1	4686	84.76	0	0.0	1	394	50	28.14	127.51	13	0	
11	11	Bravo, DJ	2	WI	MI+	Allrounder	2200	86	2004	81.39	142	34.1	0	839	70	27.97	127.12	38	1338	
12	12	Chanderpaul, S	3	WI	RCB	Batsman	9918	9	8778	70.74	14	52.8	1	25	16	8.33	80.64	0	0	
13	13	Chawla, PP	1	IND	KXIP	Allrounder	5	3	38	65.51	32	41.0	0	337	24	13.48	113.09	9	1819	
14	14	de Villiers, AB	2	SA	DD+	W. Keeper	5457	2	4998	93.19	0	0.0	1	1302	105	34.26	128.53	42	0	
15	15	Dhawan, S	2	IND	MI+	Batsman	0	0	69	56.09	0	0.0	0	1540	95	31.43	122.32	36	66	
16	16	Dhoni, MS	2	IND	CSK	W. Keeper	3509	0	6773	88.19	1	12.0	1	1782	70	37.13	136.45	64	0	
17	17	Dilshan, TM	2	SL	DD+	Allrounder	4722	32	6455	86.80	67	58.3	1	1077	76	28.34	117.83	24	356	
18	18	Dinda, AB	2	IND	KKR+	Bowler	0	0	18	60.00	5	61.4	0	6	2	1.00	33.33	0	926	
19	19	Dravid, RS	3	IND	RCB+	Batsman	13288	1	10889	71.24	4	46.5	1	1703	75	27.92	116.88	23	0	
20	20	Duminy, J-P	2	SA	MI+	Batsman	654	11	2536	84.00	25	47.6	0	978	74	36.22	119.27	35	377	
21	21	Edwards, FH	2	WI	DC	Bowler	380	157	73	45.62	60	35.6	0	4	3	4.00	80.0	0	154	
22	22	Fernando, CRD	2	SL	MI	Bowler	249	97	239	60.96	187	34.7	0	4	2	0.00	133.33	0	298	
23	23	Fleming, SP	3	NZ	CSK	Batsman	7172	0	8037	71.49	1	29.0	1	196	45	21.77	118.78	3	0	
24	24	Flintoff, A	2	ENG	CSK	Allrounder	3845	226	3394	88.82	169	33.2	1	62	24	31.00	116.98	2	105	
25	25	Gambhir, G	2	IND	DD+	Batsman	3712	0	4819	86.17	0	0.0	1	2065	93	33.31	128.90	32	0	
26	26	Ganguly, SC	3	IND	KKR+	Batsman	7212	32	11363	73.70	100	45.6	1	1349	91	25.45	106.81	42	363	

119.27

Showing 1 to 26 of 130 entries, 26 total columns

PLAYER NAME

Player name who have playing in IPL(Indian premier league).

AGE

Age of the player at the time of auction classified into 3 categories. Category 1 (L25) means the player is less than 25 years old, 2 means that the age is between 25 and 35 years ($B_{25\text{-}35}$)and category 3 means that the age is more than 35 (A_{35}).

COUNTRY

Country of origin of the player (AUS: Australia; IND: India; PAK: Pakistan; SA: South Africa; SL: Sri Lanka; NZ: New Zealand; WI: West Indies; OTH: Other countries)

TEAM

Team(s) for which the player had played in the IPL (CSK: Chennai Super Kings; DC: Deccan Chargers; DD: Delhi Daredevils; KXI: Kings XI Punjab; KKR: Kolkata Knight Riders; MI: Mumbai Indians; PWI: Pune Warriors India; RR: Rajasthan Royals; RCB: Royal Challengers Bangalore). A + sign was used to indicate that the player had played for more than one team. For example, CSK+ would mean that the player had played for CSK as well as for one or more other teams.

PLAYER ROLE

Player's primary skill (batsman, bowler, or all-rounder).

T-RUNS

Runs scored in Test matches

T-WKTS

Wickets taken in Test matches

ODI-RUNS-S

Runs scored in One Day Internationals

ODI-SR-B

Batting strike rate in One Day Internationals

ODI-WKTS

Wickets taken in One Day Internationals

ODI-SR-BL

Bowling strike rate in One Day Internationals

CAPTAINCY EXP

Captained either an T20 team or a national team

RUNS.S

Number of runs scored by a player

HS

Highest score by a batsman in IPL

AVE

Average runs scored by a batsman

SR.B

Batting strike rate (ratio of the number of runs scored to the number of balls faced) in IPL

SIXERS

Number of six runs scored by a player in IPL

RUNS.C

Number of runs conceded by a player

WKTS

Number of wickets taken by a player in IPL

AVE.BL

Bowling average (Number of runs conceded / number of wickets taken) in IPL.

ECON

Economy rate of a bowler (number of runs conceded by the bowler per over) in IPL

SR.BL

Bowling strike rate (ratio of the number of balls bowled to the number of wickets taken) in IPL

AUCTION YEAR

Year of Auction in IPL

BASE.PRICE

Player starting price in auction

SOLD.PRICE

Player sold in the auction.

2.2 DATA UNDERSTANDING

After loading the data, find str() function given information about the rows (observations) and columns (variables) along with additional information like the names of the columns.

```
> str(ipl_2013)
'data.frame':   130 obs. of  26 variables:
 $ Sl.NO.       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ PLAYER.NAME  : chr  "Abdulla, YA" "Abdur Razzak" "Agarkar, AB" "Ashwin, R" ...
 $ AGE          : int  2 2 2 1 2 2 2 2 2 2 ...
 $ COUNTRY      : chr  "SA" "BAN" "IND" "IND" ...
 $ TEAM         : chr  "KXIP" "RCB" "KKR" "CSK" ...
 $ PLAYING.ROLE : chr  "Allrounder" "Bowler" "Bowler" "Bowler" ...
 $ T.RUNS       : int  0 214 571 284 63 0 51 54 83 5515 ...
 $ T.WKTS       : int  0 18 58 31 0 0 27 50 17 1 ...
 $ ODI.RUNS.S   : int  0 657 1269 241 79 172 120 50 609 4686 ...
 $ ODI.SR.B     : num  0 71.4 80.6 84.6 45.9 ...
 $ ODI.WKTS     : int  0 185 288 51 0 0 34 62 72 0 ...
 $ ODI.SR.BL    : num  0 37.6 32.9 36.8 0 0 42.5 31.3 53 0 ...
 $ CAPTAINCY.EXP: int  0 0 0 0 0 1 0 0 1 1 ...
 $ RUNS.S       : int  0 0 167 58 1317 63 26 21 335 394 ...
 $ HS           : int  0 0 39 11 71 48 15 16 67 50 ...
 $ AVE          : num  0 0 18.6 5.8 32.9 ...
 $ SR.B         : num  0 0 121 76.3 120.7 ...
 $ SIXERS       : int  0 0 5 0 28 0 1 1 3 13 ...
 $ RUNS.C       : int  307 29 1059 1125 0 0 1342 693 610 0 ...
 $ WKTS         : int  15 0 29 49 0 0 52 37 19 0 ...
 $ AVE.BL       : num  20.5 0 36.5 23 0 ...
 $ ECON         : num  8.9 14.5 8.81 6.23 0 0 7.98 7.22 6.85 0 ...
 $ SR.BL        : num  13.9 0 24.9 22.1 0 ...
 $ AUCTION.YEAR : int  2009 2008 2008 2011 2011 2009 2011 2011 2011 2008 ...
 $ BASE.PRICE   : int  50000 50000 200000 100000 100000 50000 100000 200000 200000 200000 ...
 $ SOLD.PRICE   : int  50000 50000 350000 850000 800000 50000 500000 700000 950000 450000 ...
>
```

We can observe the output which is description of the object. It mentions that it is a list having 26 components. In the following roes,it display each one of them along with their class.

It’s is a good practice to see if there are any missing values in the data.

- null_value <- length(which(is.na(ipl_2013) ==T))
- null_value

```
> null_value <- length(which(is.na(ipl_2013)==T))
> null_value
[1] 0
>
```

The above output shows that the dataset has no missing value.

This module explains data understanding. This dataset consist of different columns. Each and

Every columns we should find the summary() function. This summary function is used to calculate the average value and determine the minimum of the columns in the data frame.

```
library(MASS)
data(ipl_2013)
summary(ipl_2013$AGE)
summary(ipl_2013$T.RUNS)
summary(ipl_2013$T.WKTS)
```

AGE

Age is numeric value which player age in times in auction. It have 3 categories the first category mean player less than 25 year’s old. second category mean player between 25 to 35 year’s old. And the third category is more than 35 year’s old. It helps to identify the player age in ipl_2013 dataset.

Summary(ipl_2013$AGE)

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.000	2.000	2.000	2.092	2.000	3.000

The below code present the result of filtering command applied on the age category. This is age value equal to the condition that display how many player in the age category.

- In the 1 category there are 16 players.
- In 2 category there are 86 players.
- In 3 category there are 28 players.

Totally 130 player records.

```
> count(filter(ipl_2013,ipl_2013$AGE==1))
   n
1 16
> count(filter(ipl_2013,ipl_2013$AGE==2))
   n
1 86
> count(filter(ipl_2013,ipl_2013$AGE==3))
   n
1 28
>
```

COUNTRY

Country is a string variable.

A player coming for which country and also playing to the same country. There are many country player in the ipl auction.

From above output, we can see the number of player in different country in ipl auction. The below R code explain the range of the column frequency for using count() function.

```
> count(ipl_2013,"COUNTRY")
   COUNTRY freq
1      AUS   22
2      BAN    1
3      ENG    3
4      IND   53
5       NZ    7
6      PAK    9
7       SA   16
8       SL   12
9       WI    6
10     ZIM    1
>
```

TEAM

Team is a string variable. Player is played to which had in IPL (Indian premier league) team. [+] sign was used to indicate that the player had played for more than one team.

From above output, we can see the number of player's are playing with ipl team. A + sign was used to indicate that the player had played for more than one team. For example, CSK+ would mean that the player had played for CSK as well as for one or more other teams. The below R code explain the range of the column frequency for using count() function.

```
> count(ipl_2013,"TEAM")
    TEAM freq
1    CSK   14
2   CSK+    5
3     DC    7
4    DC+   10
5     DD    6
6    DD+   10
7    KKR    5
8   KKR+   12
9   KXIP    5
10 KXIP+    8
11    MI    6
12   MI+    6
13   RCB    9
14  RCB+   12
15    RR    6
16   RR+    9
>
```

T.RUNS

Test Runs is an integer variable. Which contain runs of each team and bellow you can see the summary of the T.RUN also contain min, median, mean and max value and also. I took top 5 team who got high runs.

Summary(ipl_2013$T.RUN)

```
> summary(IPL2013 $ T.RUNS)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    25.5   542.5  2166.7  3002.2 15470.0
```

Top 5 runs scored players in Test matches:

```
> IPL2013 %>% arrange(desc(T.RUNS))%>% head(5) %>% select (c(2,7))
        PLAYER.NAME T.RUNS
1     Tendulkar, SR  15470
2        Dravid, RS  13288
3       Ponting, RT  13218
4        Kallis, JH  12379
5 Jayawardena, DPMD  10440
```

T. WKTS

Test wicket is an integer variable. Which contain wicket of each team and bellow you can see the summary of the T.WKTS also contain min, median, mean and max value and also I took top 5 team who got high runs.

Summary(ipl_2013$T.WKTS)

```
> summary(IPL2013 $ T.WKTS)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00    7.00   66.53   47.50  800.00
```

Top 5 wicket taked players in Test matches:

```
> IPL2013 %>% arrange(desc(T.WKTS))%>% head(5) %>% select (c(2,8))
      PLAYER.NAME T.WKTS
1 Muralitharan, M    800
2       Warne, SK    708
3       Kumble, A    619
4     McGrath, GD    563
5     Pollock, SM    421
```

ODI.RUNS. S

One day international runs score is an integer variable. It contain the scour of the one day International matches and I also filter the top 5 player who got high run in the matches and in summary class and model have null values.

Summary(ipl_2013$ODI.RUNS.S)

```
> summary(IPL2013 $ ODI.RUN.S)
Length  Class   Mode
     0   NULL   NULL
```

Top 5 runs score players one day international:

```
> IPL2013 %>% arrange(desc(ODI.RUNS.S))%>% head(5) %>% select (c(2,9))
     PLAYER.NAME ODI.RUNS.S
1  Tendulkar, SR      18426
2    Ponting, RT      13704
3 Jayasuriya, ST      13430
4     Kallis, JH      11498
5    Ganguly, SC      11363
```

ODI.SR.B

One day international strike rate ball is a number variable. It contain the strike rate of the one day international mache player. In bellow summary you can see the min, max,median and mean values and I also filter the top 5 players who get high strike rate.

Summary(ipl_2013$ODI.SR.B)

```
> summary(IPL2013 $ ODI.SR.B)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   65.65   78.22   71.16   86.79  116.66
```

Top 5 strike rate ball players in One day international:

```
> IPL2013 %>% arrange(desc(ODI.SR.B))%>% head(5) %>% select (c(2,10))
    PLAYER.NAME ODI.SR.B
1     Sharma, J   116.66
2 Shahid Afridi   113.87
3    Pathan, YK   113.60
4     Sehwag, V   104.68
5    Morkel, JA   100.25
```

ODI.WKTS

One Day International Wicket is an integer variable. Which contains number of wickets in one day international match. In bellow summary you can see the min, max, median and mean and also, I filter the top 5 player who get high wickets in the matches.

Summary(ipl_2013$ODI.WKTS)

```
> summary(IPL2013 $ ODI.WKTS)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00   18.50   76.08  106.00  534.00
```

Top 5 Wicket takers players in One Day International:

```
> IPL2013 %>% arrange(desc(ODI.WKTS))%>% head(5) %>% select (c(2,11))
      PLAYER.NAME ODI.WKTS
1 Muralitharan, M      534
2     Vaas, WPUJC      400
3     Pollock, SM      393
4     McGrath, GD      381
5          Lee, B      377
```

ODI.SR.BL

One Day International Strike Rate bowling is a number variable. Which contains number of stick rate ball in one day international match. In bellow summary you can see the min, max, median and mean and also, I filter the top 5 player who get high stick rate ball in the matches.

Summary(ipl_2013$ODI.SR.BL)

```
> summary(IPL2013 $ ODI.SR.BL)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00   36.60   34.03   45.33  150.00
```

Top 5 bowling Strike Rate players in One Day International:

```
> IPL2013 %>% arrange(desc(ODI.SR.BL))%>% head(5) %>% select (c(2,12))
    PLAYER.NAME ODI.SR.BL
1     Sharma, J     150.0
2      Kohli, V     137.0
3   Hussey, MEK     117.0
4 Henriques, MC      90.0
5   Younis Khan      86.6
```

CAPTAINCY.EXP

Captaincy Experience is an integer variable. Captaincy.exp is know the caption experience in the ipl match. Summary of the captaincy.exp

Summary(ipl_2013$CAPTAINCY.EXP)

```
> summary(IPL2013 $ CAPTAINCY.EXP)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.3154  1.0000  1.0000
```

Top 5 captaicy experience players :

```
> IPL2013 %>% arrange(desc(CAPTAINCY.EXP))%>% head(5) %>% select (c(2,13))
      PLAYER.NAME CAPTAINCY.EXP
1      Bailey, GJ             1
2        Botha, J             1
3     Boucher, MV             1
4  Chanderpaul, S             1
5 de Villiers, AB             1
```

RUNS.S

Runs's is an integer variable. Runs scored the by player by all over the ipl match. player have scored the runs in overall the ipl match by the starting for the match.

Summary(ipl_2013$RUNS.S)

```
summary(IPL2013 $ RUNS.S)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.0    39.0   172.0   514.2   925.2  2254.0
```

Top 5 runs scored players in overall the ipl match:

```
> IPL2013 %>% arrange(desc(RUNS.S))%>% head(5) %>% select (c(2,14))
    PLAYER.NAME RUNS.S
1     Raina, SK   2254
2    Gambhir, G   2065
3 Tendulkar, SR   2047
4    Sharma, RG   1975
5    Kallis, JH   1965
```

HS

High Score is an integer variable. Run scored by the indicate player in the ipl match.

It's know the with player that scored high runs in the ipl match so far.

Summary(ipl_2013$HS)

```
> summary(IPL2013 $ HS)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   16.00   35.50   47.43   73.75  158.00
```

Top 5 high score player:

```
> IPL2013 %>% arrange(desc(HS))%>% head(5) %>% select (c(2,15))
   PLAYER.NAME  HS
1 McCullum, BB 158
2    Gayle, CH 128
3    Sehwag, V 119
4   Symonds, A 117
5  Hussey, MEK 116
```

AVE

Average is a number variable. Now we can see the average of the player's in the ipl match. A player's batting average is the total number of theruns they have scored divided by the number of items they been out. Usually given to two decimal places.

Summary(ipl_2013$AVE)

```
summary(IPL2013 $ AVE)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.000   9.825  18.635  18.719  27.872  50.110
```

Top 5 batting average players :

```
> IPL2013 %>% arrange(desc(AVE))%>% head(5) %>% select (c(2,16))
    PLAYER.NAME   AVE
1     Gayle, CH 50.11
2 Pietersen, KP 42.27
3   Hussey, MEK 39.92
4 Tendulkar, SR 37.91
5     Dhoni, MS 37.13
```

SR.B:

Strike Rate batting is a numerical variable.They smash bowlers badly and increase their Strike Rate.While doing so, they also take their strike rate up.such players who scored runs with high strike rates.

Summary(ipl_2013$SR.B)

```
> summary(ipl_2013$SR.B)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   98.24  118.51  111.05  129.10  235.49
```

Top 5 strike rate palyer's in batting

```
```{r}
ipl_2013 %>% arrange(desc(SR.B)) %>% head(5) %>% select(c(2,17))

```
```

Description: df [5 x 2]

	PLAYER.NAME <chr>	SR.B <dbl>
1	Patel, MM	235.49
2	Umar Gul	205.26
3	Shahid Afridi	176.08
4	Sehwag, V	167.32
5	Bollinger, DE	165.88

5 rows

SIXERS

Sixers is a numerical variable. Six runs are scored if the ball does not bounce before passing over the boundary in the air, Its contains the six runs.

Summary(ipl_2013$SIXRES)

```
> summary(ipl_2013$SIXERS)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    1.00    6.00   17.69   29.75  129.00
```

Top 5 most sixes player's

```
```{r}
ipl_2013 %>% arrange(desc(SIXERS)) %>% head(5) %>% select(c(2,18))

```
```

Description: df [5 x 2]

	PLAYER.NAME <chr>	SIXERS <int>
1	Gayle, CH	129
2	Raina, SK	97
3	Gilchrist, AC	86
4	Sharma, RG	82
5	Pathan, YK	81

5 rows

RUNS.C

Runs conceded numerical variable. It's is the average number runs conceded per wicket taken. A bowler with an average of having a bad day in a match. It was flat track and he had already conceded in the overs.

Summary(ipl_2013$SRUNS.C)

```
> summary(ipl_2013$RUNS.C)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0   297.0   475.5   689.2  1975.0
```

Top 5 runs conceded player's

```
```{r}
ipl_2013 %>% arrange(desc(RUNS.C)) %>% head(5) %>% select(c(2,19))
```
```

Description: df [5 x 2]

	PLAYER.NAME <chr>	RUNS.C <int>
1	Pathan, IK	1975
2	Kumar, P	1919
3	Morkel, JA	1899
4	Singh, RP	1892
5	Chawla, PP	1819

5 rows

WKTS

The based bowler has taken wickets for the ipl match. The bowler want to bow the over for taking wicker for the team. The bowler can bow the only 4 over per match in ipl.

Summary(ipl_2013$WKTS)

```
> summary(ipl_2013$WKTS)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00    8.50   17.17   23.75   83.00
```

Top 5 wickers taker bowling player’s

```
```{r}
ipl_2013 %>% arrange(desc(WKTS)) %>% head(5) %>% select(c(2,20))
```
```

Description: df [5 x 2]

	PLAYER.NAME <chr>	WKTS <int>
1	Malinga, SL	83
2	Mishra, A	74
3	Singh, RP	74
4	Chawla, PP	73
5	Patel, MM	70

5 rows

AVE.BL

A player's bowling average is the number of runs they have conceded per wicket taken. The lower the bowling average is, the better the bowler is performing. It is one of a number of statistics used to compare bowlers, commonly used alongside the economy rate and the strike rate to judge the overall performance of a bowler. When a bowler has taken only a small number of wickets, their bowling average can be artificially high or low, and unstable.

Summary(ipl_2013$AVE.BL)

```
> summary(ipl_2013$AVE.BL)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00   24.79   23.11   35.58  126.30
```

Top 5 bowling average player’s

```
```{r}
ipl_2013 %>% arrange(desc(AVE.BL)) %>% head(5) %>% select(c(2,21))
```
```

Description: df [5 x 2]

	PLAYER.NAME <chr>	AVE.BL <dbl>
1	Chawla, PP	126.30
2	Kohli, V	86.25
3	Mithun, A	72.50
4	Dilshan, TM	71.20
5	Hussey, DJ	57.50

5 rows

ECON

The best career economy rate. A player's economy rate is the average number of runs they have conceded per over bowled. ... It is one of a number of statistics used to compare bowlers, commonly used alongside bowling average and strike rate to judge the overall performance of a bowler.

Summary(ipl_2013$AVE.BL)

```
> summary(ipl_2013$ECON)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   0.000   7.380   6.204   8.248  38.110 
```

Top 5 bowling average player's

```
```{r}
ipl_2013 %>% arrange(desc(ECON)) %>% head(5) %>% select(c(2,22))
```
```

Description: df [5 x 2]

	PLAYER.NAME <chr>	ECON <dbl>
1	Chawla, PP	38.11
2	Silva, LPC	21.00
3	Abdur Razzak	14.50
4	White, CL	14.00
5	Taylor, LRPL	12.00

5 rows

SR.BL

Its is a numerical variable. Bowling strike rate is measure of how quickly a bowler achieves the primary goals of bowling namely taking wickets.

Summary(ipl_2013$AVE.BL)

```
> summary(ipl_2013$SR.BL)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00    0.00   19.93   17.38   26.21  100.20 
```

Top 5 bowling average player's

```
```{r}
ipl_2013 %>% arrange(desc(SR.BL)) %>% head(5) %>% select(c(2,23))

```
```

Description: df [5 x 2]

	PLAYER.NAME <chr>	SR.BL <dbl>
1	Chawla, PP	100.20
2	Kohli, V	58.50
3	Dilshan, TM	53.00
4	Mithun, A	44.00
5	Kartik, M	41.33

5 rows

AUCTION YEAR

In IPL Auction rules as per the mega auction rules a team can use the RTM card to retain a player at the mega auction by paying the exact amount to equal the winning bid as per the IPL mega auction rules the is one of the two methods used by an IPL team to retain its star players who is the dangerous team in IPL.

Summary(ipl_2013$ACTION YEAR)

```
> summary(ipl_2013$AUCTION.YEAR)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2008    2008    2008    2009    2011    2011
```

BASE.PRICE

How is the base price of players decided for the IPL Acuctions the base price of a player is the price at which team start bidding for them it ranges between 20 lacs to 2 croes.

Summary(ipl_2013$BASE.PRICE)

```
> summary(ipl_2013$BASE.PRICE)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  20000  100000  200000  192231  225000 1350000
```

Top 5 base price player's

```
```{r}
ipl_2013 %>% arrange(desc(BASE.PRICE)) %>% head(5) %>% select(c(2,25))

```
```

Description: df [5 x 2]

	PLAYER.NAME <chr>	BASE.PRICE <int>
1	Pietersen, KP	1350000
2	Flintoff, A	950000
3	Warne, SK	450000
4	Dhoni, MS	400000
5	Dravid, RS	400000

5 rows

SOLD.PRICE

Players are bought at an auction after teams bid for them. The bidding starts at their base price and eventually the team with the highest bid gets to buy the player for the highest bid.

Summary(ipl_2013$SOLD.PRICE)

```
> summary(ipl_2013$SOLD.PRICE)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  20000  225000  437500  521223  700000 1800000
```

Top 5 most expensive sold price player's

```
```{r}
ipl_2013 %>% arrange(desc(SOLD.PRICE)) %>% head(5) %>% select(c(2,26))
```
```

Description: df [5 x 2]

	PLAYER.NAME <chr>	SOLD.PRICE <int>
1	Kohli, V	1800000
2	Sehwag, V	1800000
3	Tendulkar, SR	1800000
4	Yuvraj Singh	1800000
5	Tiwary, SS	1600000

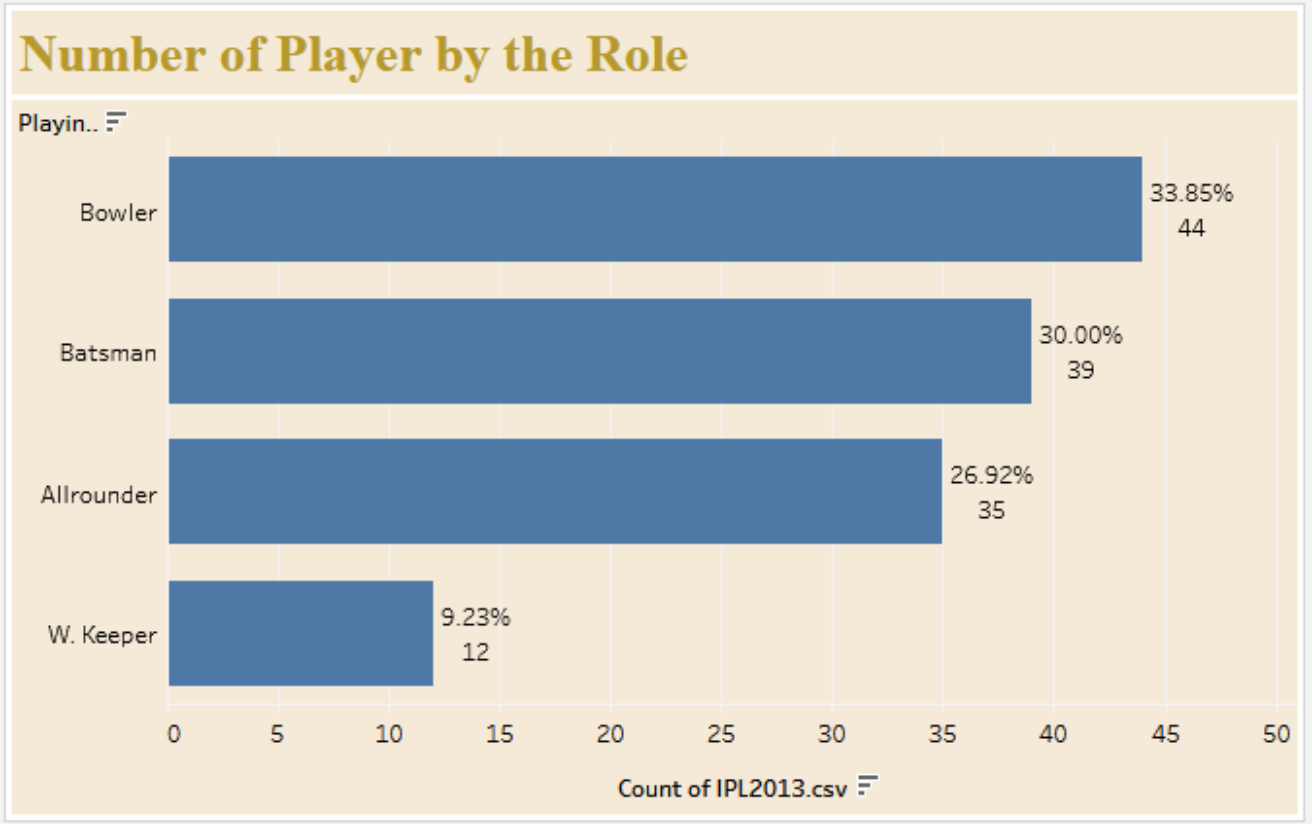
5 rows

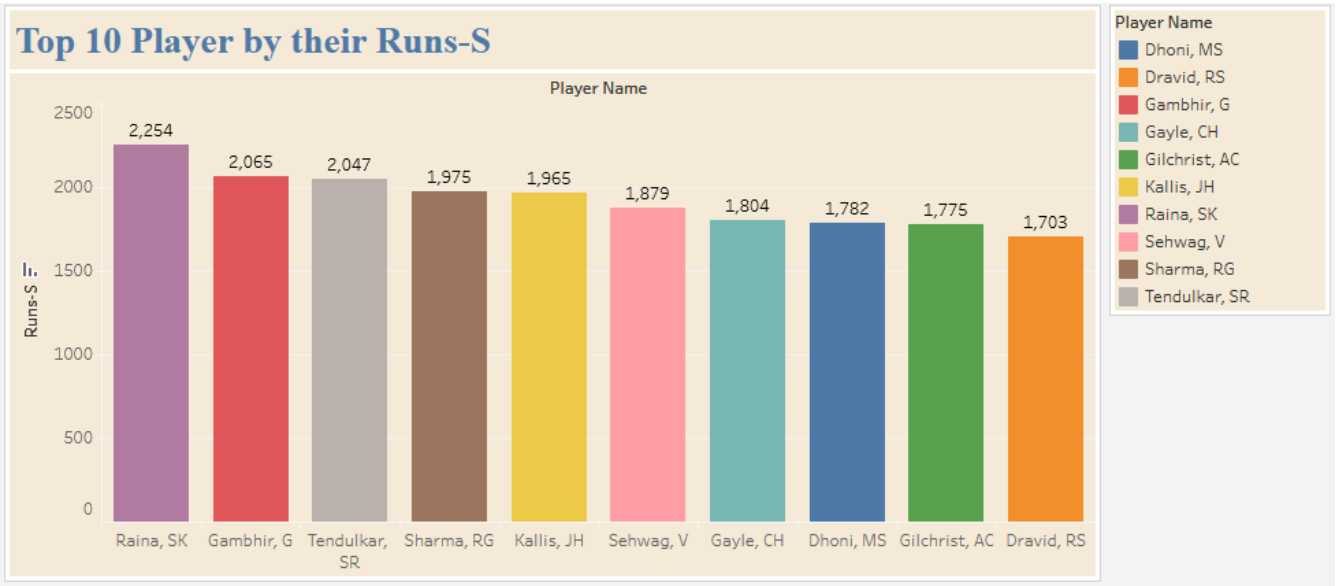
CHAPTER III: EXPLORATORY DATA ANALYSIS

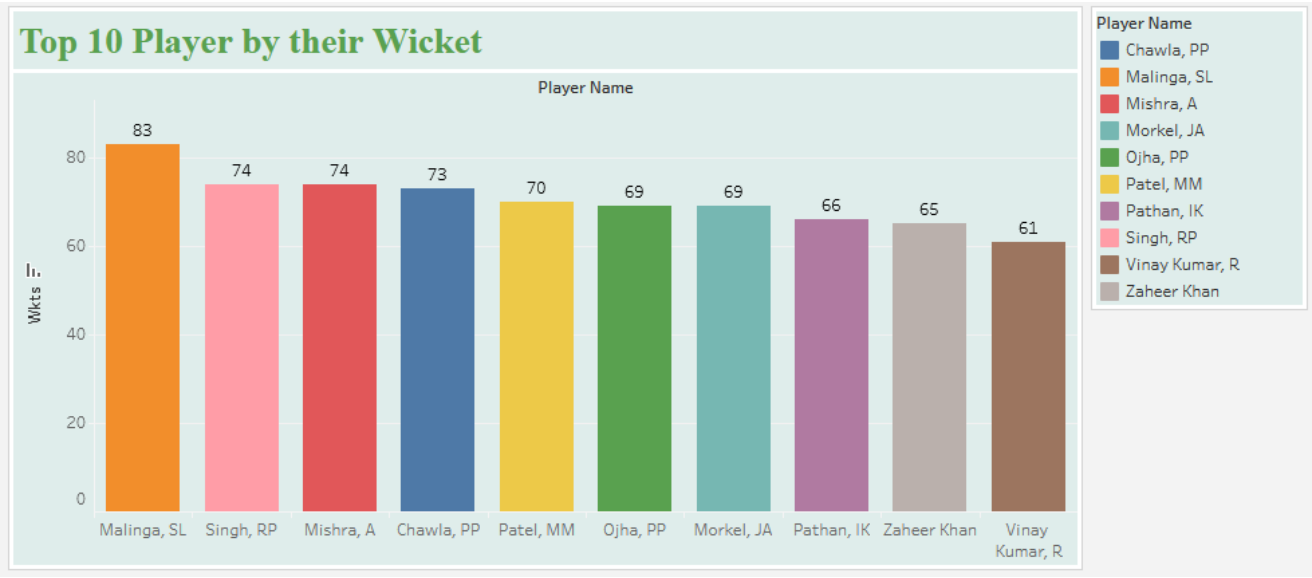
3.1 EXPLORATORY DATA ANALYSIS

Exploratory data analysis (**EDA**) is used to analyze and investigate **data** sets and summarize their main characteristics, often employing **data** visualization methods. It can also help to determine if the statistical techniques that are considering for **data** analysis are appropriate.
To see the relationship between the variables, execute pairs() command in R,

Let's see the relation between all the numerical variables.


Number of Player by the Role
Playin..
Bowler
Batsman
Allrounder
W. Keeper
33.85%
44
30.00%
39
26.92%
35
9.23%
12
0
5
10
15
20
25
30
35
40
45
50
Count of IPL2013.csv


Top 10 Player by their Runs-S
Player Name
Runs-S
2500
2000
1500
1000
500
0
2,254
2,065
2,047
1,975
1,965
1,879
1,804
1,782
1,775
1,703
Raina, SK
Gambhir, G
Tendulkar, SR
Sharma, RG
Kallis, JH
Sehwag, V
Gayle, CH
Dhoni, MS
Gilchrist, AC
Dravid, RS
Player Name
Dhoni, MS
Dravid, RS
Gambhir, G
Gayle, CH
Gilchrist, AC
Kallis, JH
Raina, SK
Sehwag, V
Sharma, RG
Tendulkar, SR


Top 10 Player by their Wicket
Player Name
Wkts
0
20
40
60
80
83
74
74
73
70
69
69
66
65
61
Malinga, SL
Singh, RP
Mishra, A
Chawla, PP
Patel, MM
Ojha, PP
Morkel, JA
Pathan, IK
Zaheer Khan
Vinay Kumar, R
Player Name
Chawla, PP
Malinga, SL
Mishra, A
Morkel, JA
Ojha, PP
Patel, MM
Pathan, IK
Singh, RP
Vinay Kumar, R
Zaheer Khan

CHAPTER IV: MODEL BUILDING

4.1 ALGORITHM

Tree based on (Random Forest and Booting) is a machine learning algorithm on supervised learning. It performs a regression task. A regression method is used where the target values is known. Regression algorithm is used for estimating the values of target as functions of predictors while building the training process. The relationship between the target and predictors is summarized and can be applied to different data set in which the target values are known. These models can be tested by the various statics that measure the difference between accepted and predicted values. The procedure gets executed in the respective dataset and the result is divided into two processes, Actual and Predicted items. Actual salary refer in the dataset and predicted represents the execution of the algorithm and its driven results.

4.2 TRAINING AND TESTING DATASET

This project concern with the IPL 2013 dataset. The goal of the data set is to predict the salary rate of individuals based on different independent variables. Split the data set into training set and testing set before the model building, The 70% data will be split into training set and 30 % data will be split into testing set.

Analysis Services randomly samples the data to help ensure that the testing and training sets are similar. By using similar data for training and testing, it can minimize the effects of data discrepancies and better understand the characteristics of the model. After a model has been processed by using the training set, test the model by making predictions against the test set. Because the data in the testing set already contains known values for the attribute that to predict, it is easy to determine whether the model's guesses are correct.

As per the universal rule, the following R code split the entire dataset into 2 portions named training data with 70 % and test data with 30 % randomly.

```r
## split the dataset into train and test
train_data <- sample(nrow(ipl_2013),nrow(ipl_2013)*0.7)
train_data
test_data <- ipl_2013[-train_data,]
test_data
```

4.3 MODEL

4.3.1 Random Forest

Random Forest When learning a technical concept, I find it's better to start with a high-level overview and work your way down into the details rather than starting at the bottom and getting immediately lost. Along those lines, this post will use an intuitive example to provide a conceptual framework of the random forest, a powerful machine learning algorithm. After getting a basic idea down, I move on to a simple implementation to see how the technique works and if it will be useful to me before finally working out the details by digging deep into the theory.

Once the dataset splitter into training and test dataset, build a model with training dataset. The following R code has been implemented the linear regression model and predict the target variable Median value.

In this analysis, build three different models for predicting the target variable sales. For the first model consider all features as predictors and pass the training data set. By analyzing the summary of model it is observed that how the predictors and target variables are related.

Now let's try with the complete Random forest based on all variables .

```{r}
library(randomForest)
set.seed(1)
random_model = randomForest(SOLD.PRICE~.,data=ipl_2013,mtry=12,importance = TRUE )
random_model
summary(random_model)
```

```
Call:
 randomForest(formula = SOLD.PRICE ~ ., data = ipl_2013, mtry = 12,      importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 12

          Mean of squared residuals: 105410820712
                    % Var explained: 35.81
                Length Class  Mode
call              5    -none- call
type              1    -none- character
predicted       130    -none- numeric
mse             500    -none- numeric
rsq             500    -none- numeric
oob.times       130    -none- numeric
importance       50    -none- numeric
importanceSD     25    -none- numeric
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           11    -none- list
coefs             0    -none- NULL
y               130    -none- numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call
```

```{r}
err = mean((test_data$SOLD.PRICE-predict.random)^2)
tss = mean((test_data$SOLD.PRICE-mean(test_data$SOLD.PRICE))^2)
rss = 1 - err/tss
rss
```

```
[1] 0.8573132
```

Based on the results of this random forest model the RSS **(residual sum of squares)** implies the model have been explains 85% of the variance.

4.3.2 Boosting

Boosting is a two-step approach, where one first uses subsets of the original data to produce a series of averagely performing models and then "boosts" their performance by combining them together using a particular cost function. Unlike bagging, in the classical boosting the subset creation is not random and depends upon the performance of the previous models: every new subsets contains the elements that were (likely to be) misclassified by previous models.

```r
library (gbm)
library(caret)

ipl_2013$PLAYER.NAME <- as.factor(ipl_2013$PLAYER.NAME)
ipl_2013$COUNTRY <-as.factor(ipl_2013$COUNTRY)
ipl_2013$TEAM <- as.factor(ipl_2013$TEAM)
ipl_2013$PLAYING.ROLE <- as.factor(ipl_2013$PLAYING.ROLE)
```

```r
model_boot <- train(SOLD.PRICE ~ .,data = ipl_2013,method = 'gbm',verbose = FALSE)
model_boot
```

```
Stochastic Gradient Boosting

130 samples
 25 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 130, 130, 130, 130, 130, 130, ...
Resampling results across tuning parameters:

  interaction.depth  n.trees  RMSE      Rsquared   MAE
  1                   50      344887.4  0.3552749  260709.0
  1                  100      333156.3  0.3882738  250761.4
  1                  150      330139.5  0.3954332  250717.6
  2                   50      334657.8  0.3811204  252594.3
  2                  100      325860.9  0.4054264  247063.6
  2                  150      324378.6  0.4095064  247471.5
  3                   50      337258.3  0.3721136  253919.8
  3                  100      328566.7  0.3993817  249108.8
  3                  150      326644.6  0.4048124  248838.8

Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was held constant at a value of 10
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 150, interaction.depth = 2, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
err.boot = mean((test_data$SOLD.PRICE-predict.boot)^2)
tss.boot = mean((test_data$SOLD.PRICE-mean(test_data$SOLD.PRICE))^2)
rss.boot = 1 - err.boot/tss.boot
rss.boot
```

```
[1] 0.842435
```

Based on the results of this Booting model the RSS **(residual sum of squares)** implies the model have been explains 84% of the variance.

This chapter built 2 different model with help of Random Forest and Booting and assess its summary with various aspects.

CHAPTER V: EVALUATION OF MODEL

5.1 Model Evaluation

Evaluating machine learning algorithm is an essential part of any project. The model may give satisfying results when evaluated using a metric accuracy score but may give poor results when evaluated against other metrics such as logarithmic loss or any other such metric. The performance measure is the way to evaluate a solution to the problem. It is the measurement that will make of the predictions made by a trained model on the test dataset. Performance measures are typically specialized to the class of problem that are working with, for example classification, regression, and clustering. Many standard performance measures will give a score that is meaningful to the problem domain. For example, classification accuracy for classification (total correct correction divided by the total predictions made multiple by 100 to turn it into a percentage).

Since this project is related to regression model, the commonly used performance measure is mean squared error (MSE). In statistics, the mean squared error (MSE) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate. Before evaluate the test error, apply the Built model to the test data.

RANDOM FOREST

Predict

The below code predicts the salary price for test data by using the best model that has been selected in the previous chapter. Now, let's look at the first few values of prediction.

```
```{r}
predict.random = predict(bag.ipl,test_data)
predict.random
```
```

```
       1        3        4        5        8       10       11       12       15       16       27       36       41       46       47       49       52
118954.9 474333.3 708947.7 656863.5 585508.2 440693.2 513641.5 270424.0 467879.8 1299534.2 864205.0 498292.5 196494.8 235065.6 124914.4 488813.2 758278.2
      53       55       56       57       58       62       63       64       76       77       80       91       96       98      108      113      115
483108.3 348034.8 709681.2 264396.2 387369.3 179045.9 408201.0 192212.9 471840.3 530262.6 351572.8 247374.8 724769.7 751537.5 279010.3 578677.6 236251.1
     116      117      122      123      130
213327.3 756076.0 450677.2 663731.2 272739.2
```

Range

Range function in R returns a vector containing the minimum and maximum of predictions produced for test data.

```{r}
range(predict.random)
```

```
[1]  118954.9 1299534.2
```

The model built in the analysis predicts value of salary price varies 118954.9 to 1299534.2

MSE

Mean squared error is an estimator measures the average of the squares of the errors that is the average squared difference between the estimated value and actual value.

```{r}
1-0.8573132
```

```
[1] 0.1426868
```

BOOSTING

Predict

The below code predicts the salary price for test data by using the best model that has been selected in the previous chapter. Now, let's look at the first few values of prediction.

```{r}
predict.boot = predict(model_boot,test_data)
predict.boot
```

```
 [1]   67283.36  462553.86  720399.16  685071.32  600452.30  444118.27  476654.89   30585.38  439265.86 1281615.09  873550.42  465738.28  169253.38  248832.76  263752.67
[16]  431366.48  871931.91  491978.68  388948.67  790850.40  314424.81  392366.51  118680.59  496698.00  229383.28  402335.40  622168.72  362522.22  379063.17  790076.67
[31]  657374.96  239549.43  582729.26  263851.52  181773.69  959890.97  385184.79  639620.70  239218.64
```

Range

Range function in R returns a vector containing the minimum and maximum of predictions produced for test data.

```{r}
range(predict.boot)
```

```
[1]    30585.38 1281615.09
```

The model built in the analysis predicts value of salary price varies 30585.38 to 1281615.09

MSE

Mean squared error is an estimator measures the average of the squares of the errors that is the average squared difference between the estimated value and actual value.

```{r}
1-0.842435
```

```
[1] 0.157565
```

CHAPTER VI: CONCLUSION

The built model predicts the player salary for the dataset named IPL 2013, with the following conclusions.

- The MSE value of Random Forest model is 14%.
- The MSE value of Boosting model is 15%.
- Comparing the two values, MSE value of Random Forest error rate is smaller than error rate in this model.
- So, we consider random forest model to predict the ipl player salary in future.