NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA

INTRODUCTION TO MACHINE LEARNING(18CSE751)
LA

on

IPL 1st Inning Score Prediction using Machine Learning

Submitted in partial fulfilment of the requirement for the award of Degreeof

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by:

NALIN RAJ R	1NT18CS105
VIGNESH M	1NT18CS089
KUMARA H	1NT18CS081

Department of Computer Science and Engineering

(Accredited by NBA Tier-1)

2021-2022

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM
, APPROVED BY AICTE & GOVT.OF KARNATAKA)

Department of Computer Science and Engineering

(Accredited by NBA Tier-1)

CERTIFICATE

This is to certify that the Project work titled **"IPL 1st Inning Score Prediction using Machine Learning"** is an authentic work carried out by Nalin Raj R(1NT18CS105), VIGNESH M(1NT18CS089),KUMARA H(1NT18CS081)bonafide students of Nitte Meenakshi Institute of Technology, Bangalore in partial fulfilment for the award of the degree of *Bachelor of Engineering* in COMPUTER SCIENCE AND ENGINEERING of Visvesvaraya Technological University, Belagavi during the academic year *2021-2022*. It is certified that all corrections and suggestions indicated during the internal assessment has been incorporated in the report.

S

Signature of internal guide	**Signature of the HOD**
Dr.Vani V Professor-Dept CSE, NMIT Bangalore	Dr.Sarojadevi H HOD- Dept. CSE, NMIT Bangalore

DECLARATION

We hereby declare that

(i) The project work is our original work

(ii) This Project work has not been submitted for the award of any degree or examination at any other university/College/Institute.

(iii) This Project Work does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.

(iv) This Project Work does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

a) their words have been re-written but the general information attributed to them has been referenced;

b) where their exact words have been used, their writing has been placed inside quotation marks, and referenced.

(v) This Project Work does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

NAME	USN	Signature
NALIN RAJ R	1NT18CS105	
VIGNESH M	1NT18CS089	
KUMARA H	1NT18CS081	

Date: 17/01/2022

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success. I express my sincere gratitude to our Principal **Dr. H. C. Nagaraj**, Nitte Meenakshi Institute of Technology for providing facilities.

We wish to thank our HoD**, Dr. Sarojadevi H.** for the excellent environment created to further educational growth in our college. We also thank him for the invaluable guidance provided which has helped in the creation of a better project.

I hereby like to thank our **DR. Vani V**, professor, Computer science department for Incubation, Innovation, Research and Consultancy on his periodic inspection, time to time evaluation of the project and help to bring the project to the present form.

Thanks to our Departmental Project coordinators. We also thank all our friends, teaching and non-teaching staff at NMIT, Bangalore, for all the direct and indirect help provided in the completion of the project.

Signature

NALIN RAJ R	1NT18CS105	
VIGNESH M	1NT18CS089	
KUMARA H	1NT18CS081	

Date: 17/01/2022

ABSTRACT

Cricket is one of the most celebrated games in the world. With the introduction of machine learning techniques in the world of cricket, predicting the score of the match has been established as one of the most challenging problems. We will use some of these factors to predict score using machine learning algorithms. This Machine Learning model adapts a Regression Approach to predict the score of the First Inning of an IPL Match. Supervised machine learning is the construction of algorithms that are able to produce general patterns and hypotheses by using externally supplied instances to predict the fate of future instances. Supervised machine learning classification algorithms aim at categorizing data from prior information. Classification is carried out very frequently in data science problems

In the last decade a large number of supervised learning methods have been introduced in the field of the machine learning. Supervised learning became an area for a lot of research activity in machine learning. Many of the supervised learning techniques have found application in their processing and analyzing variety of data. One of the main characteristics is that the supervised learning has the ability of annotated training data. The so called labels are class labels in the classification process. There is a variety of algorithms that are used in the supervised learning method.

TABLE OF CONTENTS

INTRODUCTION

The Dataset contains ball by ball information of the matches played between IPL Teams of **Season 1 to 10**, i.e. from 2008 to 2017. This Machine Learning model adapts a Regression Appoach to predict the score of the First Inning of an IPL Match. Cricket is being played in many countries around the world. There are a lot of domestic and international cricket tournaments being held in many countries. The cricket game has various forms such as Test Matches, Twenty20 Internationals, Internationals one day, etc. IPL is also one of them, and has great popularity among them. It's a twenty-20 cricket game league played to inspire young and talented players in India. The league was conducted annually in March, April or May and has a huge fan base among India. There are eight teams which represent eight cities which are chosen from an auction. These teams compete against each other for the trophy. The whole match depends on the luck for the team, player's performance and lot more parameters that will be taken in to the consideration. The match that is played before the day is also will make a change in the prediction. The stakeholders are much more benefited due to the huge popularity and the huge presence of people at the venue. The accuracy of a data depends on the size of the data we take for analysing and the records that are taken for predicting the outcome.

Cricket is a game played between two teams comprising of 11 players in each team. The result is either a win, loss or a tie. However, sometimes due to bad weather conditions the game is also washed out as Cricket is a game which cannot be played in rain. Moreover, this game is also extremely unpredictable because at every stage of the game the momentum shifts to one of the teams between the two. A lot of times the result gets decided on the last ball of the match where the game gets really close. Considering all these unpredictable scenarios of this unpredictable game, there is a huge interest among the spectators to do some prediction either at the start of the game or during the game. Many spectators also play betting games to win money. There can be several factors that strongly affect predictions like the current score, wickets in hand, weather conditions, dew factor, pitch condition, etc. We have used a data set of 1,79,079 records consisting of the data for every single ball in IPL matches from the year 2009 to 2019.My work develops some crucial predictions using machine learning model Linear regressor.

BACKGROUND

FEATURE CONSTRUCTION: The existing features like over, ball,is super over,wide runs,non-stricker, etc. are not good enough to make confident, reliable predictions for a final score, Therefore new features score are created as follows:

• Balls remaining: number of balls remaining in the first innings of the match

• Current score: current score of the team.

• Wickets remaining: wickets in hand for the team.

• Final score: final score of that team in that match this is the target variable which we are trying to forecast.

When these newly created features are used to predict the final score, we obtained some handsome value of R square for different machine learning models used in the project.

PREDICTING FIRST INNINGS SCORES: Score prediction for the first innings is a typical multiple regression problem since the output is the forecasted score. Data set of size 10,315 records is used to train our model.

System Configuration

3.1 Hardware System Configuration:

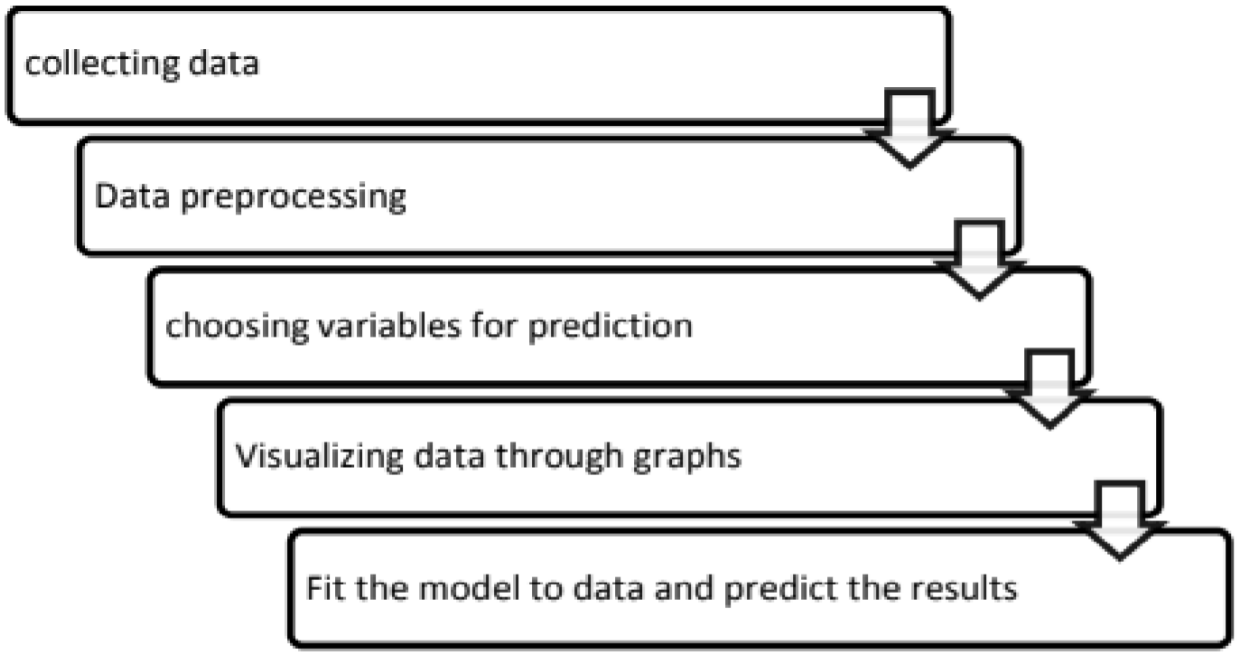
- Processor - Pentium – IV
- Speed - 1.1 GHz
- RAM - 256 MB (min)
- Hard Disk - 20 GB

3.2 Software System Configuration:

- Operating System - XP/7/8/8.1/10
- Coding Language - Python 3 and above version Hardware Requirements
- Processors - Pentium IV Processor
- Speed - 3.00 GHZ
- RAM - 2 GB
- Storage - 20 GB

Approach and Design

The below figure explains the approach we have taken into building the predictive model using machine learning algorithms.



4.1 Data Collection

Data collection is the process of gathering and measuring information from countless different sources. In order to use the data, we collect to develop practical machine learning solutions.
Collecting data allows you to capture a record of past events so that we can use data analysis to find recurring patterns. From those patterns, you build predictive models using machine learning algorithms that look for trends and predict future changes.

The Indian Premier League's official website is the principal basis of data for this project. The data was web scrapped from the website and kept in the appropriate format using a python library called beautiful soup. The dataset has the columns regarding match-number, IPL season year, the place where match has been held and the stadium name, the match winner details, participating teams, the margin of winning and the umpire details, player of the match etc. Indian Premier League was only 11 years old, which is why, after the pre-processing, only 577 matches were available. Here, some of the columns may contain null values and some of the attributes may not be required for match winner prediction which is discussed in data preprocessing.

4.2 Data Preprocessing

4.2.1 Data cleaning

There are some null values in the dataset in the columns such as winner, city, venue etc. Due to the presence of these null values, the classification cannot be done accurately. So, we tried to replace the null values in different columns with dummy values.

4.2.2 Choosing Required Attributes

This step is the main part where we can eliminate some columns of the dataset that are not useful for the estimation of match winning team. This is estimated using feature importance. The considered attributes have the following feature importance

Implementation

5.1 Import Necessary Libraries

In [39]:
```
# Importing Necessary Libraries
import pandas as pd
import numpy as np
np.__version__
```

Out[39]: '1.18.5'

Mounting GDrive for importing Dataset

Mount your Google Drive and save the dataset in the Drive name "data.csv"

In [41]:
```
# Mounting GDrive and importing dataset
data = pd.read_csv('/content/drive/My Drive/data.csv')
print(f"Dataset successfully Imported of Shape : {data.shape}")
```

```
Dataset successfully Imported of Shape : (76014, 15)
```

Exploratory Data Analysis

In [42]:
```
# First 5 Columns Data
data.head()
```

Out[42]:

	mid	date	venue	batting_team	bowling_team	batsman	bowler	runs	wickets	overs	runs_last_5	wickets_last_5	striker	non-striker	total
0	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	SC Ganguly	P Kumar	1	0	0.1	1	0	0	0	222
1	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	1	0	0.2	1	0	0	0	222
2	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.2	2	0	0	0	222
3	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.3	2	0	0	0	222
4	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.4	2	0	0	0	222

Numerical Values Of The Dataset

In [43]:
```
# Describing Numerical Values of the Dataset
data.describe()
```

Out[43]:

	mid	runs	wickets	overs	runs_last_5	wickets_last_5	striker	non-striker	total
count	76014.000000	76014.000000	76014.000000	76014.000000	76014.000000	76014.000000	76014.000000	76014.000000	76014.000000
mean	308.627740	74.889349	2.415844	9.783068	33.216434	1.120307	24.962283	8.869287	160.901452
std	178.156878	48.823327	2.015207	5.772587	14.914174	1.053343	20.079752	10.795742	29.246231
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	67.000000
25%	154.000000	34.000000	1.000000	4.600000	24.000000	0.000000	10.000000	1.000000	142.000000
50%	308.000000	70.000000	2.000000	9.600000	34.000000	1.000000	20.000000	5.000000	162.000000
75%	463.000000	111.000000	4.000000	14.600000	43.000000	2.000000	35.000000	13.000000	181.000000
max	617.000000	263.000000	10.000000	19.600000	113.000000	7.000000	175.000000	109.000000	263.000000

Information and Unique Values in Each Column

In [44]:
```
# Information (not-null count and data type) About Each Column
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76014 entries, 0 to 76013
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   mid             76014 non-null  int64  
 1   date            76014 non-null  object 
 2   venue           76014 non-null  object 
 3   batting_team    76014 non-null  object 
 4   bowling_team    76014 non-null  object 
 5   batsman         76014 non-null  object 
 6   bowler          76014 non-null  object 
 7   runs            76014 non-null  int64  
 8   wickets         76014 non-null  int64  
 9   overs           76014 non-null  float64
 10  runs_last_5     76014 non-null  int64  
 11  wickets_last_5  76014 non-null  int64  
 12  striker         76014 non-null  int64  
 13  non-striker     76014 non-null  int64  
 14  total           76014 non-null  int64  
dtypes: float64(1), int64(8), object(6)
memory usage: 8.7+ MB
```

In [45]:
```
# Number of Unique Values in each column
data.nunique()
```

Out[45]:

```
mid               617
date              442
venue              35
batting_team       14
bowling_team       14
batsman           411
bowler            329
runs              252
wickets            11
overs             140
runs_last_5       102
wickets_last_5      8
striker           155
non-striker        88
total             138
dtype: int64
```

Datatypes of All Columns

In [46]:
```
# Datatypes of all Columns
data.dtypes
```

Out[46]:
```
mid                int64
date              object
venue             object
batting_team      object
bowling_team      object
batsman           object
bowler            object
runs               int64
wickets            int64
overs            float64
runs_last_5        int64
wickets_last_5     int64
striker            int64
non-striker        int64
total              int64
dtype: object
```

5.2 Data Cleaning

Removing irrelevant data from our dataset

In [47]:
```
# Names of all columns
data.columns
```

Out[47]:
```
Index(['mid', 'date', 'venue', 'batting_team', 'bowling_team', 'batsman',
       'bowler', 'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5',
       'striker', 'non-striker', 'total'],
      dtype='object')
```

Here, we can see that columns *['mid', 'date', 'venue', 'batsman', 'bowler', 'striker', 'non-striker']* won't provide any relevant information for our model to train

In [48]:
```
irrelevant = ['mid', 'date', 'venue','batsman', 'bowler', 'striker', 'non-striker']
print(f'Before Removing Irrelevant Columns : {data.shape}')
data = data.drop(irrelevant, axis=1) # Drop Irrelevant Columns
print(f'After Removing Irrelevant Columns : {data.shape}')
data.head()
```

```
Before Removing Irrelevant Columns : (76014, 15)
After Removing Irrelevant Columns : (76014, 8)
```

Out[48]:

	batting_team	bowling_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
0	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.1	1	0	222
1	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.2	1	0	222
2	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.2	2	0	222
3	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.3	2	0	222
4	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.4	2	0	222

Keeping only consistent teams

(Teams never change in current season)

In [49]:
```
# Define Consistent Teams
const_teams = ['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',
               'Mumbai Indians', 'Kings XI Punjab', 'Royal Challengers Bangalore',
               'Delhi Daredevils', 'Sunrisers Hyderabad']
```

In [50]:
```
print(f'Before Removing Inconsistent Teams : {data.shape}')
data = data[(data['batting_team'].isin(const_teams)) & (data['bowling_team'].isin(const_teams))]
print(f'After Removing Irrelevant Columns : {data.shape}')
print(f"Consistent Teams : \n{data['batting_team'].unique()}")
data.head()
```

```
Before Removing Inconsistent Teams : (76014, 8)
After Removing Irrelevant Columns : (53811, 8)
Consistent Teams :
['Kolkata Knight Riders' 'Chennai Super Kings' 'Rajasthan Royals'
 'Mumbai Indians' 'Kings XI Punjab' 'Royal Challengers Bangalore'
 'Delhi Daredevils' 'Sunrisers Hyderabad']
```

Out[50]:

	batting_team	bowling_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
0	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.1	1	0	222
1	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.2	1	0	222
2	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.2	2	0	222
3	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.3	2	0	222
4	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.4	2	0	222

5.3 Data Preprocessing and Encoding

Performing Label Encoding

In [53]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
for col in ['batting_team', 'bowling_team']:
  data[col] = le.fit_transform(data[col])
data.head()
```

Out[53]:

	batting_team	bowling_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
32	3	6	61	0	5.1	59	0	222
33	3	6	61	1	5.2	59	1	222
34	3	6	61	1	5.3	59	1	222
35	3	6	61	1	5.4	59	1	222
36	3	6	61	1	5.5	58	1	222

Performing one Hot Coding and Column Transforming

In [54]:
```
from sklearn.compose import ColumnTransformer
columnTransformer = ColumnTransformer([('encoder',
                                        OneHotEncoder(),
                                        [0, 1])],
                                      remainder='passthrough')
```

In [55]:
```
data = np.array(columnTransformer.fit_transform(data))
```

Save the Numpy Array in a new DataFrame with transformed columns

In [56]:
```
cols = ['batting_team_Chennai Super Kings', 'batting_team_Delhi Daredevils', 'batting_team_Kings XI Punjab',
              'batting_team_Kolkata Knight Riders', 'batting_team_Mumbai Indians', 'batting_team_Rajasthan Royals',
              'batting_team_Royal Challengers Bangalore', 'batting_team_Sunrisers Hyderabad',
              'bowling_team_Chennai Super Kings', 'bowling_team_Delhi Daredevils', 'bowling_team_Kings XI Punjab',
              'bowling_team_Kolkata Knight Riders', 'bowling_team_Mumbai Indians', 'bowling_team_Rajasthan Royals',
              'bowling_team_Royal Challengers Bangalore', 'bowling_team_Sunrisers Hyderabad', 'runs', 'wickets', 'overs',
       'runs_last_5', 'wickets_last_5', 'total']
df = pd.DataFrame(data, columns=cols)
```

In [57]:
```
# Visualize Encoded Data
df.head()
```

Out[57]:

	batting_team_Chennai Super Kings	batting_team_Delhi Daredevils	batting_team_Kings XI Punjab	batting_team_Kolkata Knight Riders	batting_team_Mumbai Indians	batting_team_Rajasthan Royals	batting_team_Royal Challengers Bangalore	batting_team_Sunrisers Hyderabad
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0

5.4 Model Building

Prepare Train and Test Splits

In [58]:
```
features = df.drop(['total'], axis=1)
labels = df['total']
```

In [59]:
```
# Perform 80 : 20 Train-Test split
from sklearn.model_selection import train_test_split
train_features, test_features, train_labels, test_labels = train_test_split(features, labels, test_size=0.20, shuffle=True)
print(f"Training Set : {train_features.shape}\nTesting Set : {test_features.shape}")
```

```
Training Set : (32086, 21)
Testing Set : (8022, 21)
```

5.5 Model Algorithms

Decision Tree Regression

1. Decision Tree Regressor

In [61]:
```
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor()
# Train Model
tree.fit(train_features, train_labels)
```

Out[61]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

In [62]:
```
# Evaluate Model
train_score_tree = str(tree.score(train_features, train_labels) * 100)
test_score_tree = str(tree.score(test_features, test_labels) * 100)
print(f'Train Score : {train_score_tree[:5]}%\nTest Score : {test_score_tree[:5]}%')
models["tree"] = test_score_tree
```

```
Train Score : 99.98%
Test Score : 86.13%
```

In [63]:
```
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as mse
print("---- Decision Tree Regressor - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(test_labels, tree.predict(test_features))))
print("Mean Squared Error (MSE): {}".format(mse(test_labels, tree.predict(test_features))))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(test_labels, tree.predict(test_features)))))
```

```
---- Decision Tree Regressor - Model Evaluation ----
Mean Absolute Error (MAE): 3.9665918723510347
Mean Squared Error (MSE): 122.64298180004987
Root Mean Squared Error (RMSE): 11.0744291861951
```

Linear Regression

Linear Regression

In [64]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
# Train Model
linreg.fit(train_features, train_labels)
```

Out[64]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [65]:
```
# Evaluate Model
train_score_linreg = str(linreg.score(train_features, train_labels) * 100)
test_score_linreg = str(linreg.score(test_features, test_labels) * 100)
print(f'Train Score : {train_score_linreg[:5]}%\nTest Score : {test_score_linreg[:5]}%')
models["linreg"] = test_score_linreg
```

```
Train Score : 65.91%
Test Score : 65.91%
```

In [66]:
```
print("---- Linear Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(test_labels, linreg.predict(test_features))))
print("Mean Squared Error (MSE): {}".format(mse(test_labels, linreg.predict(test_features))))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(test_labels, linreg.predict(test_features)))))
```

```
---- Linear Regression - Model Evaluation ----
Mean Absolute Error (MAE): 13.095521348645773
Mean Squared Error (MSE): 301.4403800998092
Root Mean Squared Error (RMSE): 17.36203847766181
```

Support Vector Machine Regression

Support Vector Machine

In [73]:
```
from sklearn.svm import SVR
svm = SVR()
# Train Model
svm.fit(train_features, train_labels)
```

Out[73]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [74]:
```
train_score_svm = str(svm.score(train_features, train_labels)*100)
test_score_svm = str(svm.score(test_features, test_labels)*100)
print(f'Train Score : {train_score_svm[:5]}%\nTest Score : {test_score_svm[:5]}%')
models["svm"] = test_score_svm
```

```
Train Score : 57.48%
Test Score : 57.45%
```

In [75]:
```
print("---- Support Vector Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(test_labels, svm.predict(test_features))))
print("Mean Squared Error (MSE): {}".format(mse(test_labels, svm.predict(test_features))))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(test_labels, svm.predict(test_features)))))
```

```
---- Support Vector Regression - Model Evaluation ----
Mean Absolute Error (MAE): 14.689273833142883
Mean Squared Error (MSE): 376.2689686154565
Root Mean Squared Error (RMSE): 19.39765368840924
```

CONCLUSION

- Successfully implemented different classification methods.
- On comparison of various methods, we found that some of the models are precise whereas some are accurate.
- Also, we found some drawbacks for each implemented method.

REFERENCES

- https://www.kaggle.com/yuvrajdagur/ipl-dataset-season-2008-to-2017
- https:// ardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2tps://tow
- https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2
- https://www.datacamp.com/community/tutorials/decision-tree-classification-python
- https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning
- https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/
- https://www.javatpoint.com/data-preprocessing-machine-learning
- https://docs.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model#:~:text=A%20machine%20learning%20model%20is,and%20learn%20from%20those%20data.
- https://www.w3schools.com/python/python_ml_train_test.asp

APPENDIX

1. **Dataset**

   https://www.kaggle.com/yuvrajdagur/ipl-dataset-season-2008-to-2017

2. **Code**

   https://github.com/vigneshrocks/18CSE751