Prediction of Cricket Score

**Project description:**

 A**Data science project which predicts cricket score of first innings of a team** by giving few features Problem Statement Indian Premier League (IPL) is a Twenty20 cricket format league in India. It is usually played in April and May every year. The league was founded by Board of Control for Cricket India (BCCI) in 2008.

**Project Technical Details:**

The following diagram shows the various steps that we have followed in our project.

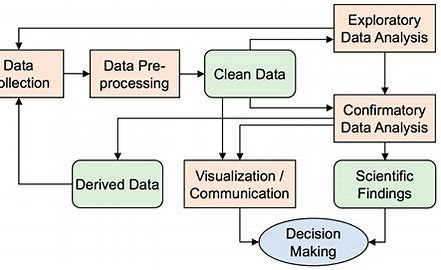


fig 1: General steps of Data analysis process



The above picture clearly tells you how bad is taking run rate as a single factor to predict the final score in an limited overs cricket match.  In ODI and T-20 cricket, many factors play a key role in deciding what the final score will be. some of the key factors:

* Number of wickets left
* Number of balls left
* On how much scores are the current batsman batting?
* How much the team had scored in last 5 overs?
* How much the team had lost wickets in last 5 overs?
* The nature of the pitch
* How strong is the batting and bowling team?

I will use some of these factors to predict score using machine learning algorithms. We use regression analysis in machine learning to predict the final score of an T-20 match.

1. ***Data collection:-***

*Collected the below details related to the factors influencing the account dormancy through conducting various surveys and documented the details. The dataset included both categorical and numerical data.*

*The data collected by Kaggle Link :* [*IPL Complete Dataset (2008-2020) | Kaggle*](https://www.kaggle.com/datasets/patrickb1912/ipl-complete-dataset-20082020)

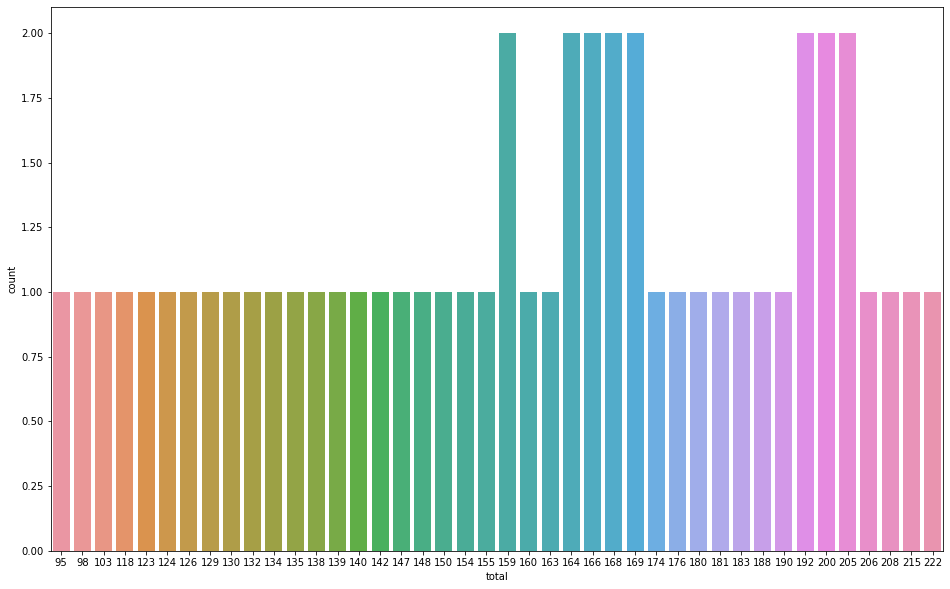
Dataset consists of following columns(features):

* mid: Each match is given a unique number
* date: When the match happened
* venue: Stadium where match is being played
* bat\_team: Batting team name
* bowl\_team: Bowling team name
* batsman: Batsman name who faced that ball
* bowler: Bowler who bowled that ball
* runs: Total runs scored by team at that instance
* wickets: Total wickets fallen at that instance
* overs: Total overs bowled at that instance
* runs\_last\_5: Total runs scored in last 5 overs
* wickets\_last\_5: Total wickets that fell in last 5 overs
* striker: max(runs scored by striker, runs scored by non-striker)
* non-striker: min(runs scored by striker, runs scored by non-striker)
* total: Total runs scored by batting team after first innings.

# *2. Exploratory Data Analysis:*

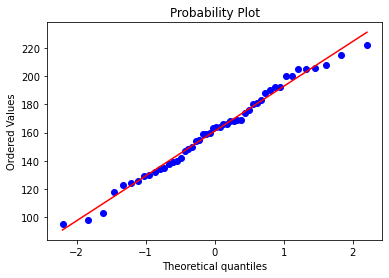
*Go through all the features and try to understand the story behind each feature. See how these features are inter-related. If you can’t understand the data, take the help of plots. They give you great understanding of the data.*

*You understand when you visualize it pictorially. Picture speaks a lot of data*





* *We check the probability plot to confirm that the residuals are indeed normally distributed.*

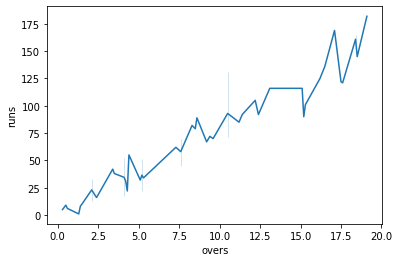


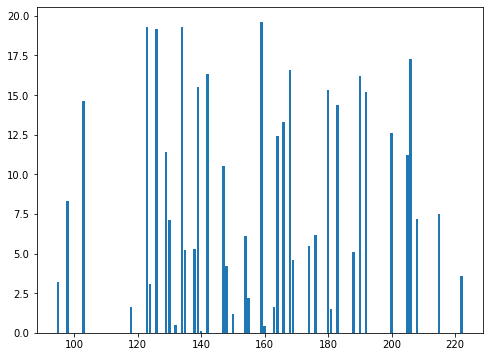
probability plot

*From the barplot, we can easily make inferences regarding the year in which a particular team has scored the maximum wins (and also the number of wins)*

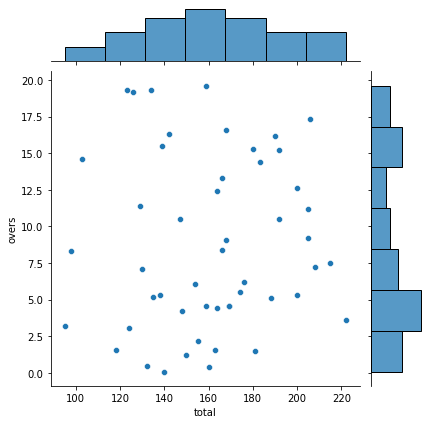
***Barplot-***

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent.

******



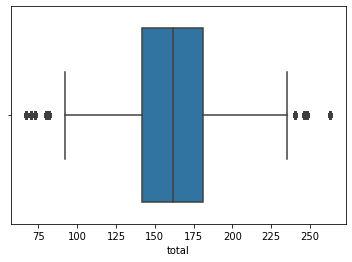
***Joint plot-***

******

*A Jointplot comprises three plots. Out of the three, one plot displays a bivariate graph which shows how the dependent variable(Y) varies with the independent variable(X). Another plot is placed horizontally at the top of the bivariate graph and it shows the distribution of the independent variable(X). The third plot is placed on the right margin of the bivariate graph with the orientation set to vertical and it shows the distribution of the dependent variable(Y).*

***Box Plot-***

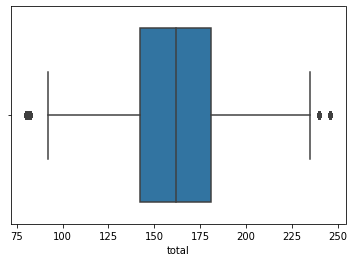
* ***It is showing to much of outliers so we can use winsorizations or trimming.***

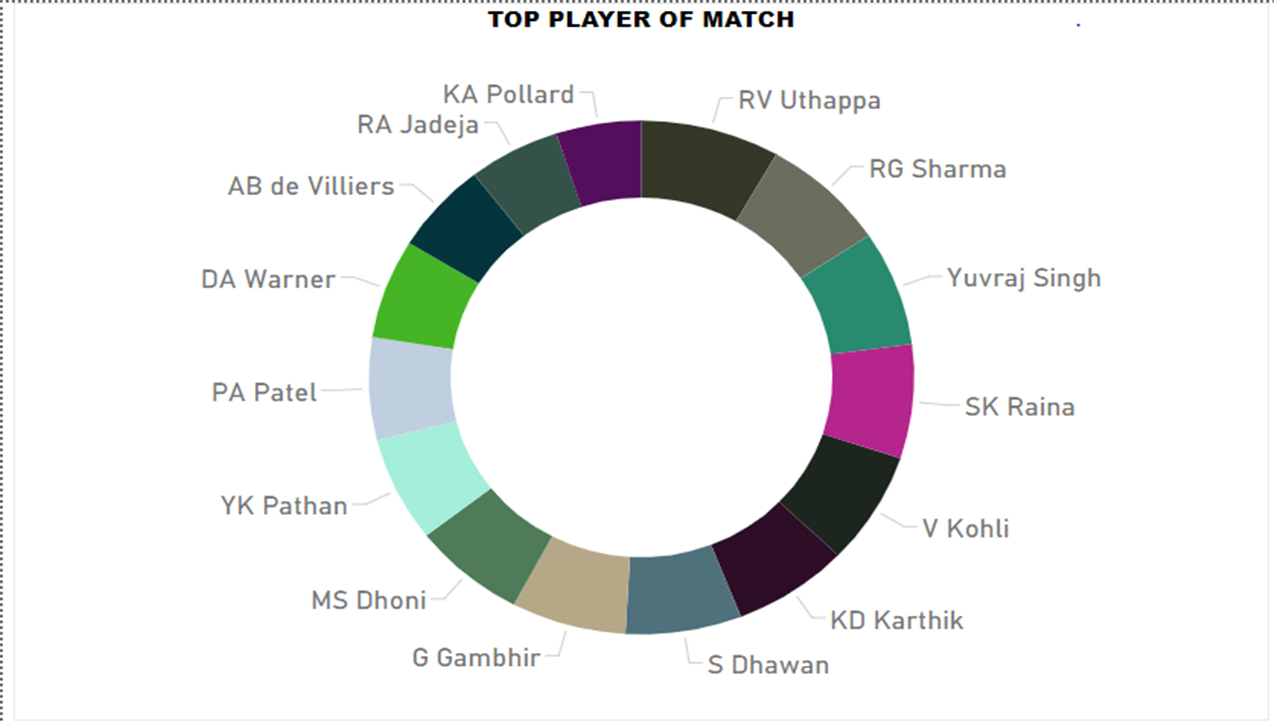
******

***Inferences from the above charts: -***

* *The .xlsx file has data of IPL matches starting from the season 2008 to 2017*
* *We have found 425 raws that are outliers*.
* *We will use trimming as dataset is too large.*

**After Trimming-**



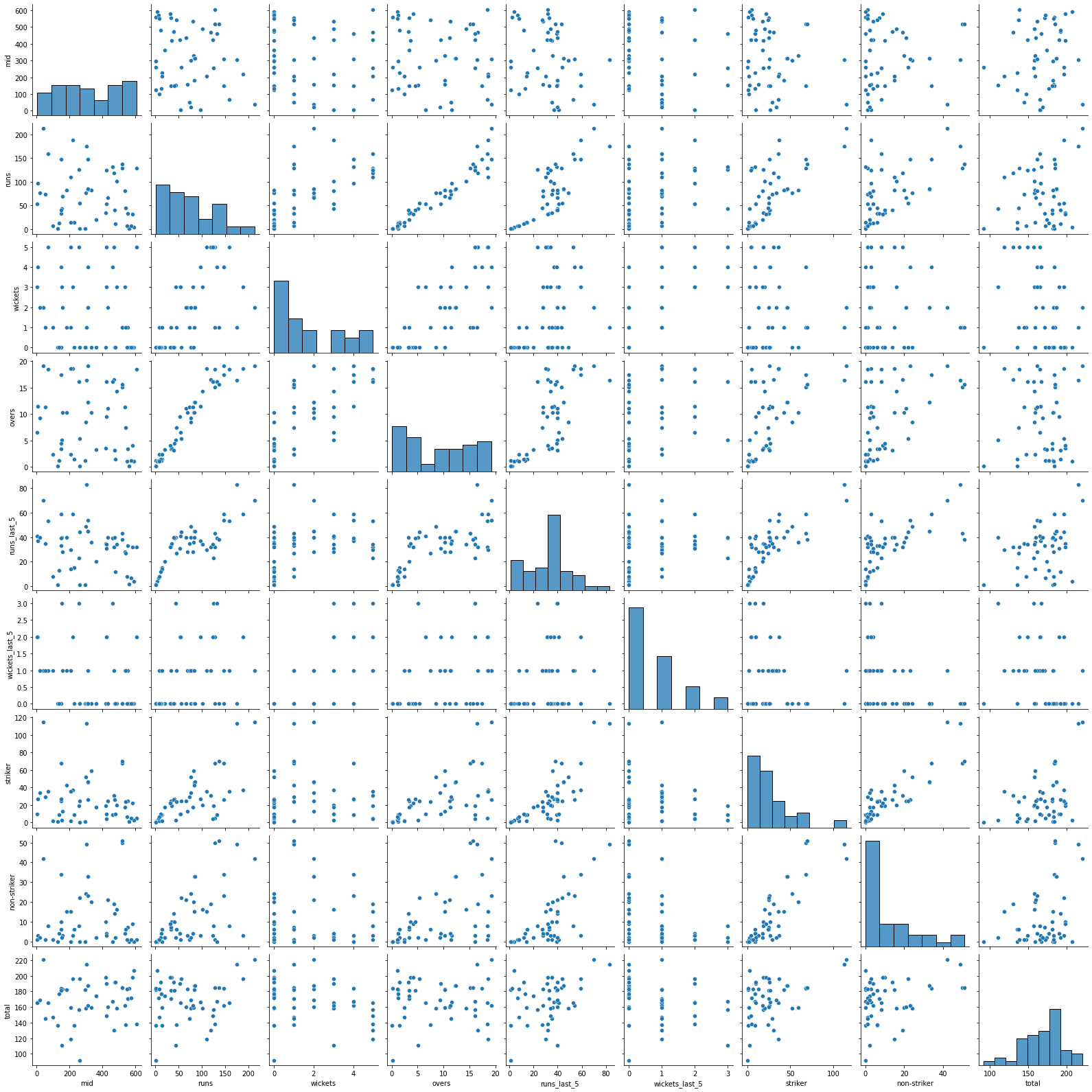


***Description of pair plot of all the variables***

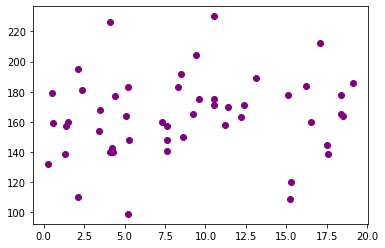
* *Pairplot visualizes given data to find the relationship between them where the variables can be continuous or categorical*

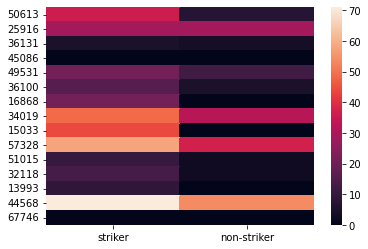
*Let’s plot data using pairplot:-*

*From the picture below, we can observe the variations in each plot. The plots are in matrix format where the row name represents x axis and column name represents the y axis.*



***Heat map & Scatter Plot-***

******



*Heatmap is the way of representing the data in a 2-dimentional form*

* all the values are less than 0 .5 so the features are not correlated.
* We can also see in scatter plot that the data is not that much correlated.

***3. Feature Engineering***

*This is the most important step in any Data Science project. Sometimes, the features are readily available. Using the visualizations, understand which features helps us to predict the class label better. If you can’t go with the given features, engineer/build new features using the domain knowledge. This involves a lot of Math and Statistics and geometric intuition*.

***Code EDA-***

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

data=pd.read\_excel('ipl.xlsx')

data.head()

# this will show starting 5 rows from the dataset

data.isnull()

#

data.notnull()

#

data.fillna(method='pad') # this method will replace null value with previous one.

# Now we will apply EDA on the given dataset

######################## First Business Moment Decision ##################################

#######################Measure of Central Tendency############################################

# mean

# mode

# median

data.mean()

data.median()

data.mode()

######################## Second Business Moment Decision ##################################

####################### Measure of Spread ############################################

# var

# std

# max

# quantile

data.var()

data.std()

print(data['total'].max())

print(data['total'].min())

data.quantile()

data.describe()

######################## Third Business Moment Decision ##################################

####################### Skewness  ########################################################

data['total'].skew()

# negatively skewed (also known as left-skewed) distribution is

# a type of distribution in which more values are concentrated on the right side (tail) of the

# distribution graph while the left tail of the distribution graph is longer

a=data.total.unique()

a

plt.hist(data.total.unique()) #histogram

######################## Fourth Business Moment Decision ##################################

####################### Kurtosis  #######################################################

data.kurt()

# Kurtosis refers to the degree of presence of outliers in the distribution.

#  Kurtosis is a statistical measure, whether the data is heavy-tailed or light-tailed in a normal distribution

sns.boxplot(data.total)

print("Highest allowed",data['total'].mean() + 3\*data['total'].std())

# maximum threshold

print("Lowest allowed",data['total'].mean() - 3\*data['total'].std())

# minimum threshold

a=data[(data['total'] <73) | (data['total'] >248)]

a.shape

# we Will use trimming as dataset is too large

# we have found 425 raws that are outliers

print(data.shape)

df=data[(data['total'] > 75) & (data['total'] < 248)]

print(df.shape)

sns.boxplot(df.total)

temp=df.sample(50) # we make a temporary dataframe with the help of sample function

#  that halp to take random values from whole dataset.

plt.figure(1,figsize=(16,10))

sns.countplot(temp['total'] )

# frequency = countplot

plt.figure(1,figsize=(8,6))

plt.bar(temp['total'] ,temp['overs'])

sns.jointplot(x=temp['total'],y=temp['overs'])

from scipy import stats

import pylab

stats.probplot(temp.total, dist = "norm", plot = pylab)

plt.show()

sns.pairplot(temp.iloc[:, :])

sns.lineplot(data = temp, x = "overs", y = "runs")

temp1=df.sample(15)

sns.heatmap(temp1.loc[: ,["striker", "non-striker"]])

# The primary purpose of Heat Maps is to better visualize

# the volume of locations/events within a dataset and assist in directing viewers

# towards areas on data visualizations that matter most

############################### Correlation ########################################

# Correlation measures the relationship between two variables. We mentioned that

#  a function has a purpose to predict a value, by converting input (x) to output (f(x)).

#  We can say also say that a function uses the relationship between two variables for prediction

np.corrcoef(temp.total,temp.overs)

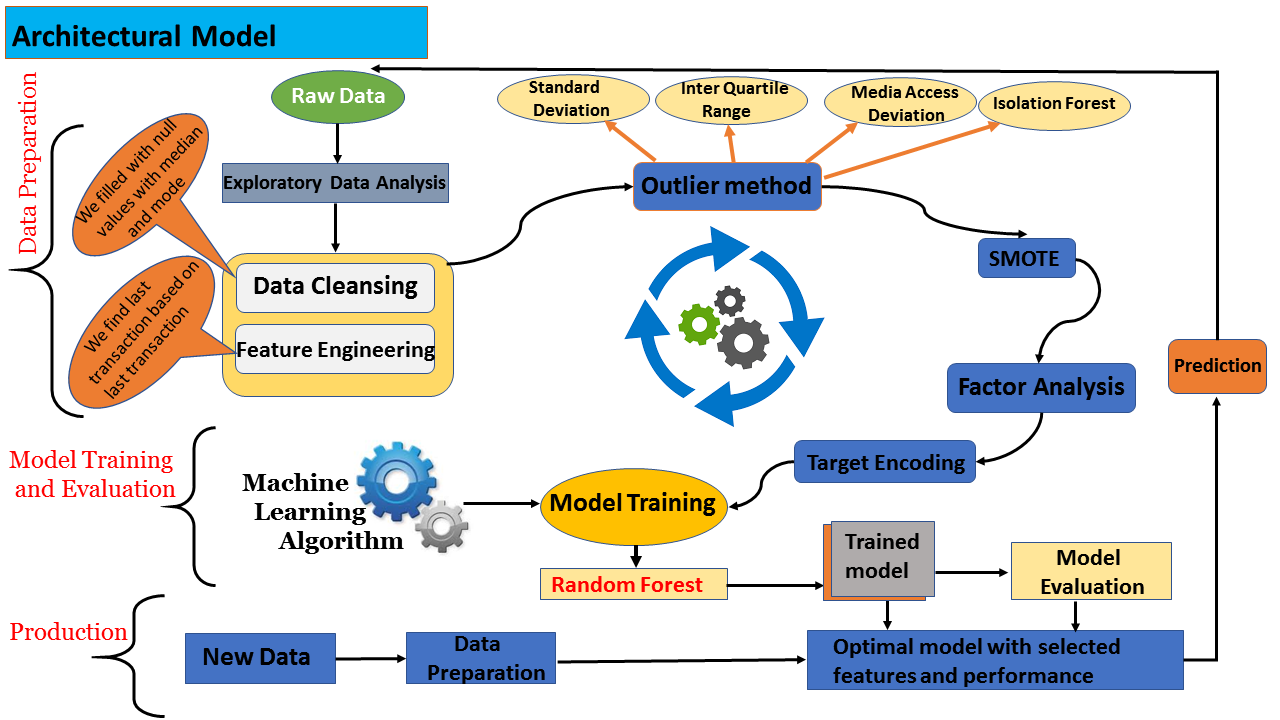
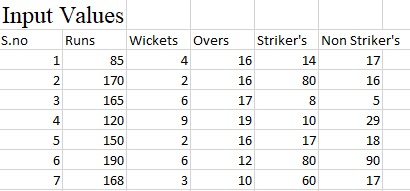
plt.scatter(y=temp.total,x=temp.overs,color='purple')

df.corr()

# we see there exists low collinearity between input variables

# so no collinearity problem

4. Model building:

**Input variable –**

Importing the dataset

import pandas as pd

dataset = pd.read\_csv('data/odi.csv')

X = dataset.iloc[:,[7,8,9,12,13]].values #Input features

y = dataset.iloc[:, 14].values #Label

I have used ‘ipl.csv’ datafile here for predicting scores in T20 Cricket.

**Features Used:**

* runs
* wickets
* overs
* striker
* non-striker

**Why didn’t I use other features?**

While experimenting, all the other features didn’t make much difference in results. You can use a different combination of features and test the code on them.

**Label Used**: Total

## Splitting data into training and testing set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

We will train our model on 75 percent of the dataset and test the model on remaining dataset.

## Feature Scaling the data

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

## Training the dataset

* Using Linear Regression Algorithm
* from sklearn.linear\_model import LinearRegression
* lin = LinearRegression()
* lin.fit(X\_train,y\_train)
* Using Random Forest Regression Algorithm
* from sklearn.ensemble import RandomForestRegressor
* lin = RandomForestRegressor(n\_estimators=100,max\_features=None)
* lin.fit(X\_train,y\_train)

You can use any one of these algorithms but as you will see later random forest regression gives us better accuracy.

## Testing the dataset on trained model

y\_pred = lin.predict(X\_test)

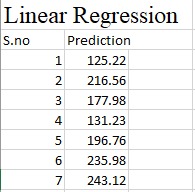
score = lin.score(X\_test,y\_test)\*100

print("R-squared value:" , score)

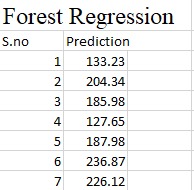
R-squared value

R-sqaured is a statistic that will give some information about the goodness of fit of a model. In regression, the R-squared coefficient of determination is a statistical measure of how well the regression predictions approximate the real data points. An R-squared value of 1 indicates that the regression predictions perfectly fit the data.

Linear regression-



Random Forest Regression-



**R squared value –**

* **linear regression-(60.4)**
* **Random Forest regression-(71.9)**

***Selected model: Random Forest***

Using Lazy classifier techniques various machine learning algorithms were explored such as KNN, Ada Boost, Decision Tree, ensemble techniques , the model that gave the highest accuracy for this data set is Random Forest.

import pandas as pd

dataset = pd.read\_excel('ipl.xlsx')

X = dataset.iloc[:,[7,8,9,12,13]].values

y = dataset.iloc[:, 14].values

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Training the dataset

from sklearn.ensemble import RandomForestRegressor

reg = RandomForestRegressor(n\_estimators=100,max\_features=None)

reg.fit(X\_train,y\_train)

import pickle

pickle.dump(reg,open("model.pkl","wb"))

import os

os.getcwd()

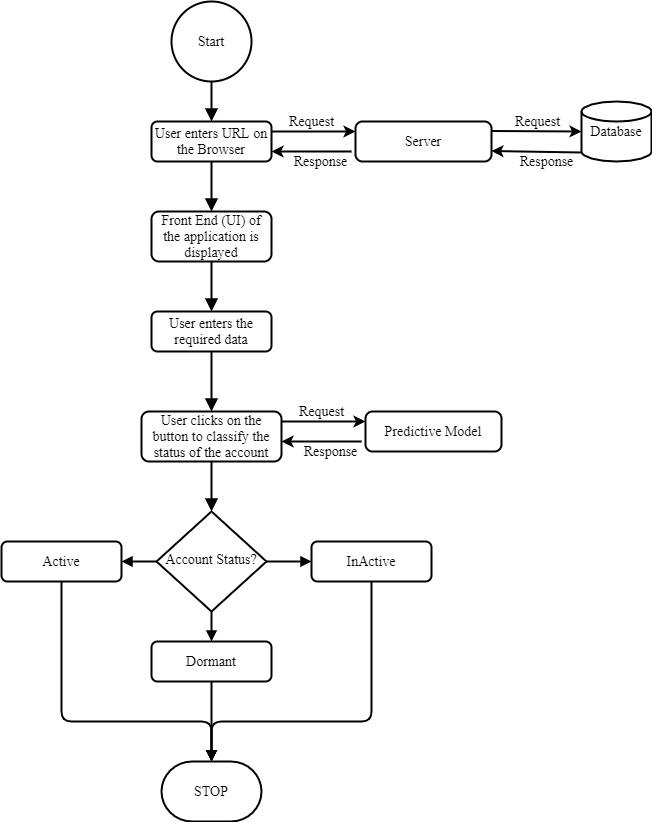
**Cricket Score prediction uses the following packages and library from python:**

|  |
| --- |
| **import pandas as pd**  **import matplotlib.pyplot as plt**  **import seaborn as sns**  **import numpy as np**  **import sklearn**  **from sklearn.ensemble import RandomForestClassifier**  **from dataprep.eda import plot,plot\_correlation,plot\_missing,create\_report**  **from sklearn.preprocessing import StandardScaler** |

# 6. Deployment using Flask:

Deployment process was done using flask technique.

**Deployment Architecture**

****

Created app.py file to show prediction of Cricket Score-

|  |
| --- |
|  |