# A Project/Dissertation Review Report

on

## IPL SCORE PREDICTION USING MACHINE LEARNING

***Submitted in partial fulfillment of the***

***requirement for the award of the degree of***

COMPUTER SCIENCE AND ENGINEERING
WITH SPECIALISATION IN DATA
ANALYTICS


**UnderTheSupervision of**

**Mr.Vetrivendan L.**
**Associate Professor**

Submitted By
**ABHINAV GANGWAR (18SCSE1120013)**
**AYUSH KUMAR (18SCSE1120007)**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**GALGOTIAS UNIVERSITY, GREATER NOIDA**
**INDIA**
**July, 2021-2022**

## Abstract

Cricket, particularly the Twenty20 organization, has the greatest vulnerability, where a solitary over can totally change the energy of the game. With a large number of individuals following the Indian Premier League (IPL), fostering a model for anticipating the result of its matches is a genuine issue. A cricket match relies on different variables, and in this work, the elements which altogether impact the result of a Twenty20 cricket match are distinguished. Every player's presence in the field is considered to discover the general weight (relative strength) of the groups. A multivariate relapse-based arrangement is proposed to ascertain points of every player in the association and the general load of a group is figured dependent on the past presentation of the players who have shown up most for the group. At long last, a dataset was displayed dependent on the recognized seven variables which impact the result of an IPL match. Six AI models were prepared and utilized for foreseeing the result of each 2018 IPL match, 15 minutes before the interactivity, following the throw. The forecast results are great. The issues with the dataset and how the exactness of the classifier can be worked on further are talked about.

## Acronyms

| | |
|---|---|
| B.Tech. | Bachelor of Technology |
| M.Tech. | Master of Technology |
| BCA | Bachelor of Computer Applications |
| MCA | Master of Computer Applications |
| B.Sc. (CS) | Bachelor of Science in Computer Science |
| M.Sc. (CS) | Master of Science in Computer Science |
| SCSE | School of Computing Science and Engineering |

## Table of Contents

# CHAPTER-1
# INTRODUCTION

Since the beginning of the IPL in 2008, it has drawn in watchers from one side of the planet to the other. An undeniable degree of vulnerability and last second nail biters has encouraged fans to watch the matches. Inside a brief period, IPL has turned into the most elevated income producing class of cricket. In a cricket match, we frequently see the score line showing the likelihood of the group winning dependent on the current match circumstance. This expectation is normally finished with the assistance of Data Analytics. Prior to when there were no headways in AI, the forecast was typically founded on instincts or some fundamental calculations.

Amidst this pandemic, the IPL has kept us engaged and snared onto our seats. I most definitely being an ardent aficionado of the competition and the game, chosen to give my hands a shot a dataset to foresee the runs scored.

In this article I will attempt to go through every one of the pieces and rudiments of the interaction so that even a newbie in the information science local area will actually want to understand

# CHAPTER-2
# LITERATUE SURVEY

The work done on Data Mining of Cricket dataset portrays the different information mining strategies viz Decision Tree, Naive Bayes, KNN, Random Forest applied on the IPL dataset, the model is worked for foreseeing the consequences of the matches. The best ascribes were chosen utilizing the Wrapper and Ranker technique and afterward the grouping has been finished. This work was finished with the assistance that the determination of the best group is continuously needed by the administration for best result. The paper gives the ideal answer for select the best group utilizing Data Mining Techniques instead of following the conventional strategy which is drawn-out. At the point when we are proclaiming a period for the specific title it is compulsory to choose the best group thus the shot at the group to be the hero turns out to be simple. the creators propose the fluffy grouping rationale. The after effects of the IPL batting Statistics were assembled into different groups and it gave proficient and successful exact outcomes with the Data Mining Technique – Clustering. The idea of grouping is utilized in request to arrange batting insights of the Indian Premier League which has the fluffy information into suitable bunches. Raza Ul Mustafa et al introduced a review on the examination of the achievability of utilizing the Twitter information to figure the consequences of the match. The work has been proposed to check the AI methods' adequacy when applied on information gathered to get knowledge gotten from web-based media organizations also, other true occasions are anticipated. The procedures utilized in their work are Support Vector Machine, Naive Bayes Classifier furthermore, the Linear Regression. The SVM procedure holds great. Live Cricket Score and Winning Prediction work depicts with regards to the structure of the model which predicts the score for the pursuing group and will gauge the score of the second innings of match. The proposed work utilizes the ideas of Linear Relapse, Naive Bayes Classifier and

Reinforce Learning Calculation. The variables, for example, throw result, positioning of the group, host group advantage were thought of.

A Step by step approach:

- Information cleaning and designing
- Exploratory Data Analysis
- Component Engineering and Selection
- Think about Multiple Algorithms
- Perform Hyperparameter Tuning
- Assess the models
- Convey the model

Devices utilized:

- Jupyter Notebook/Google colab
- Visual Studio

Innovation utilized:

- AI.
- Carafe (Front-end incorporation).
- All things considered, for the smooth running of the task we've utilized not many libraries like NumPy, Pandas, Scikit-learn, TensorFlow, and Matplotlib.

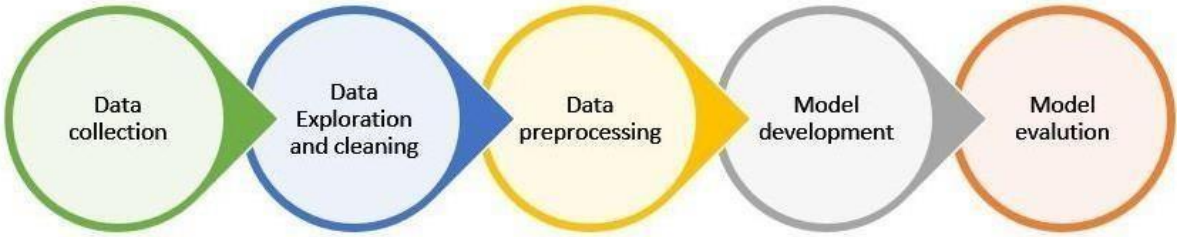



**Fig. Proposed system architecture**

## Understanding the DATASET

The dataset consists of 15 columns:

mid: The match id to uniquely identify each match.

date: The date on which the match was held.

venue: The name of the stadium.

bat_team: The batting team name.

bowl_team: The bowling team name.

batsman: The name of the batsman.

bowler: The name of the bowler.

runs: The runs scored till now.

wickets: The wickets taken till now.

overs: The number of overs bowled.

runs_last_5: The number of runs scored in last 5 overs.

wickets_last_5: The number of wickets taken in last 5 overs.

striker: The name of the batsmen on the batting end.

non-striker: The name of the batsmen on the bowling end.

total: The total number of runs scored in the match.

## Step 1: Data cleaning and formatting

In this step we will remove all the unwanted columns and clean any row for missing values. Here I have shown how to remove the unwanted columns.

```
remove_columns = ['striker', 'non-striker', 'mid', 'batsman', 'bowler']
df.drop(labels = remove_columns, axis=1, inplace=True)
```

## Step 2: Exploratory Data Analysis

Here, we will explore the data and decide what data we want to keep for feature engineering. Now that we know how many times a stadium is being used, we can choose only the top stadiums which have most data for our model prediction.

```
df['bat_team'].unique()
df['venue'].unique()

#to see most used stadiums
df['count'] = 1
df.groupby(['venue']).count()['count']
```

Output of unique teams:

```
array(['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',
       'Mumbai Indians', 'Deccan Chargers', 'Kings XI Punjab',
       'Royal Challengers Bangalore', 'Delhi Daredevils',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Sunrisers Hyderabad',
       'Rising Pune Supergiants', 'Gujarat Lions',
       'Rising Pune Supergiant'], dtype=object)
```

Output of unique venues:

```
array(['M Chinnaswamy Stadium',
       'Punjab Cricket Association Stadium, Mohali', 'Feroz Shah Kotla',
       'Wankhede Stadium', 'Eden Gardens', 'Sawai Mansingh Stadium',
       'Rajiv Gandhi International Stadium, Uppal',
       'MA Chidambaram Stadium, Chepauk', 'Dr DY Patil Sports Academy',
       'Newlands', "St George's Park", 'Kingsmead', 'SuperSport Park',
       'Buffalo Park', 'New Wanderers Stadium', 'De Beers Diamond Oval',
       'OUTsurance Oval', 'Brabourne Stadium',
       'Sardar Patel Stadium, Motera', 'Barabati Stadium',
       'Vidarbha Cricket Association Stadium, Jamtha',
       'Himachal Pradesh Cricket Association Stadium', 'Nehru Stadium',
       'Holkar Cricket Stadium',
       'Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium',
       'Subrata Roy Sahara Stadium',
       'Shaheed Veer Narayan Singh International Stadium',
       'JSCA International Stadium Complex', 'Sheikh Zayed Stadium',
       'Sharjah Cricket Stadium', 'Dubai International Cricket Stadium',
       'Maharashtra Cricket Association Stadium',
       'Punjab Cricket Association IS Bindra Stadium, Mohali',
       'Saurashtra Cricket Association Stadium', 'Green Park'],
      dtype=object)
```

Output of number of times a stadium appears in the data by using the groupby() function:

```
venue
Barabati Stadium                                  856
Brabourne Stadium                                1380
Buffalo Park                                      380
De Beers Diamond Oval                             368
Dr DY Patil Sports Academy                       2088
Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium    1113
Dubai International Cricket Stadium               868
Eden Gardens                                     7049
Feroz Shah Kotla                                 7068
Green Park                                        492
Himachal Pradesh Cricket Association Stadium     1115
Holkar Cricket Stadium                            617
JSCA International Stadium Complex                837
Kingsmead                                        1731
M Chinnaswamy Stadium                            7443
MA Chidambaram Stadium, Chepauk                  5972
Maharashtra Cricket Association Stadium          1843
Nehru Stadium                                     499
New Wanderers Stadium                             995
Newlands                                          737
OUTsurance Oval                                   251
Punjab Cricket Association IS Bindra Stadium, Mohali    1342
Punjab Cricket Association Stadium, Mohali       4247
Rajiv Gandhi International Stadium, Uppal        5827
Sardar Patel Stadium, Motera                     1484
Saurashtra Cricket Association Stadium           1229
Sawai Mansingh Stadium                           4110
Shaheed Veer Narayan Singh International Stadium     742
Sharjah Cricket Stadium                           744
Sheikh Zayed Stadium                              836
St George's Park                                  870
Subrata Roy Sahara Stadium                       2086
SuperSport Park                                  1377
Vidarbha Cricket Association Stadium, Jamtha      370
Wankhede Stadium                                 7048
Name: count, dtype: int64
```

## Step 3: Feature Engineering and Selection

In this step we will decide which teams and venues to keep for making our model. We have used only the current playing teams and their respective home grounds as of October 2020.

```
current_teams = [ Kolkata Knight Riders ,  Chennai Super Kings ,  Rajasthan Royals ,
       'Mumbai Indians', 'Kings XI Punjab',
       'Royal Challengers Bangalore', 'Delhi Daredevils',
       'Sunrisers Hyderabad']

current_venues = ['M Chinnaswamy Stadium','Eden Gardens', 'Feroz Shah Kotla', 'MA Chidambaram
       'Punjab Cricket Association Stadium, Mohali',
       'Wankhede Stadium', 'Sawai Mansingh Stadium',
       'Rajiv Gandhi International Stadium, Uppal']

df = df[(df['bat_team'].isin(current_teams)) &(df['bowl_team'].isin(current_teams))]
df = df[(df['venue'].isin(current_venues))]

# Removing the first 5 overs data in every match
df = df[df['overs']>=5.0]
```

Now we will use Data Preprocessing to convert features using One Hot Encoding and also convert the string date into datetime object.

```
# Converting 'date' from string into datetime object
from datetime import datetime
df['date'] = df['date'].apply(lambda x: datetime.strptime(x, '%m/%d/%Y'))

# --- Data Preprocessing ---
# Converting categorical features using OneHotEncoding method
final_df = pd.get_dummies(data=df, columns=['bat_team', 'bowl_team', 'venue'])
```

```
#rearranging the columns
final_df = final_df[['date','bat_team_Chennai Super Kings', 'bat_team_Delhi Daredevils',
       'bat_team_Kings XI Punjab', 'bat_team_Kolkata Knight Riders',
       'bat_team_Mumbai Indians', 'bat_team_Rajasthan Royals',
       'bat_team_Royal Challengers Bangalore', 'bat_team_Sunrisers Hyderabad',
       'bowl_team_Chennai Super Kings', 'bowl_team_Delhi Daredevils',
       'bowl_team_Kings XI Punjab', 'bowl_team_Kolkata Knight Riders',
       'bowl_team_Mumbai Indians', 'bowl_team_Rajasthan Royals',
       'bowl_team_Royal Challengers Bangalore',
       'bowl_team_Sunrisers Hyderabad', 'venue_Eden Gardens',
       'venue_Feroz Shah Kotla','venue_M Chinnaswamy Stadium',
       'venue_MA Chidambaram Stadium, Chepauk','venue_Sawai Mansingh Stadium',
       'venue_Punjab Cricket Association Stadium, Mohali',
       'venue_Rajiv Gandhi International Stadium, Uppal',
       'venue_Wankhede Stadium', 'overs','runs', 'wickets', 'runs_last_5', 'wickets_last_5',
       'total']]

final_df.head()
```

| | date | bat_team_Chennai Super Kings | bat_team_Delhi Daredevils | bat_team_Kings XI Punjab | bat_team_Kolkata Knight Riders | bat_team_Mumbai Indians | bat_team_Rajasthan Royals | bat_team_Royal Challengers Bangalore | bat_team_Sunrisers Hyderabad |
|---|---|---|---|---|---|---|---|---|---|
| 32 | 2008-04-18 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 33 | 2008-04-18 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 34 | 2008-04-18 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 35 | 2008-04-18 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 36 | 2008-04-18 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

## Step 4 ,5: Compare Multiple Algorithms & Perform Hyper-parameter Tuning

I have just compared 2 algorithms here Lasso Regression and Random Forest Regression. First we will have to divide our data into train set and test set before using a machine learning algorithm.

*Dividing into Train and Test Data:*

```
# Splitting the data into train and test set
X_train = final_df.drop(labels='total', axis=1)[final_df['date'].dt.year<=2016]
X_test = final_df.drop(labels='total', axis=1)[final_df['date'].dt.year>=2017]

y_train = final_df[final_df['date'].dt.year<=2016]['total'].values
y_test = final_df[final_df['date'].dt.year>=2017]['total'].values

# Removing the 'date' column
X_train.drop(labels='date', axis=True, inplace=True)
X_test.drop(labels='date', axis=True, inplace=True)
```

Now that we have dataset for training and testing, the first algorithm we will look at is Lasso Regression. We have used Grid Search CV for hyperparameter tuning.

```
# --- Model Building ---
# Lasso Regression Model
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV

lasso=Lasso()
parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40]}
lasso_regressor=GridSearchCV(lasso,parameters,scoring='neg_mean_squared_error',cv=5)
lasso_regressor.fit(X_train,y_train)
print(lasso_regressor.best_params_)
print(lasso_regressor.best_score_)
prediction = lasso_regressor.predict(X_test)
```

```
{'alpha': 1}
-343.1309689444174
```

Best Parameter and best score

Below is the code for Random Forest Regression. I have used Randomized Search CV for hyperparamter tuning.

```
#__Random forest regressor__
from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 800, num = 4)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 60, num = 6)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800]}
```

Output of values of random_grid

Now we will find the best parameters and fit the model to make predictions. This code will require some time to compute.

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 100,
# Fit the random search model
rf_random.fit(X_train, y_train)

print(rf_random.best_params_)
prediction = rf_random.predict(X_test)
```

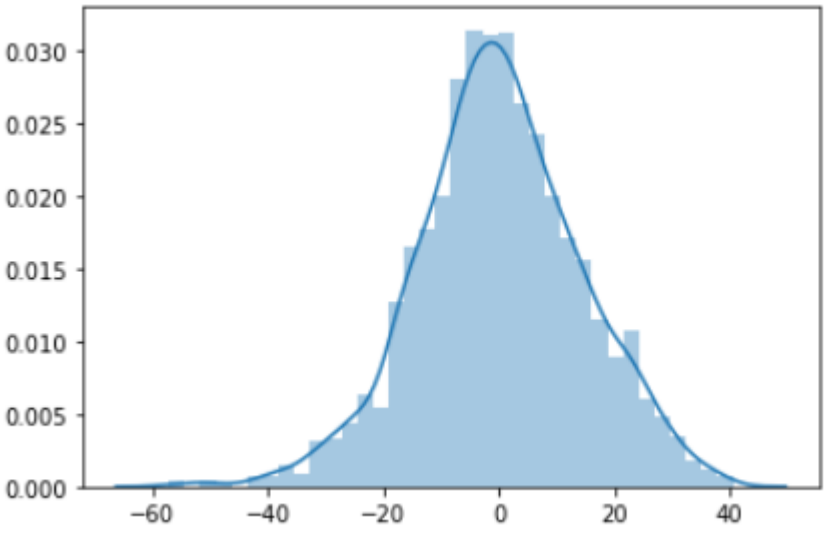
```
{'n_estimators': 600,
 'min_samples_split': 5,
 'min_samples_leaf': 4,
 'max_features': 'auto',
 'max_depth': 10,
 'bootstrap': True}
```

The best parameters from hyperparameter tuning

## Step 6: Evaluate the models

Lasso Regression Evaluating the Lasso Regression model using Distplot and Sklearn Metrics:

```
print(sns.distplot(y_test-prediction))
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```



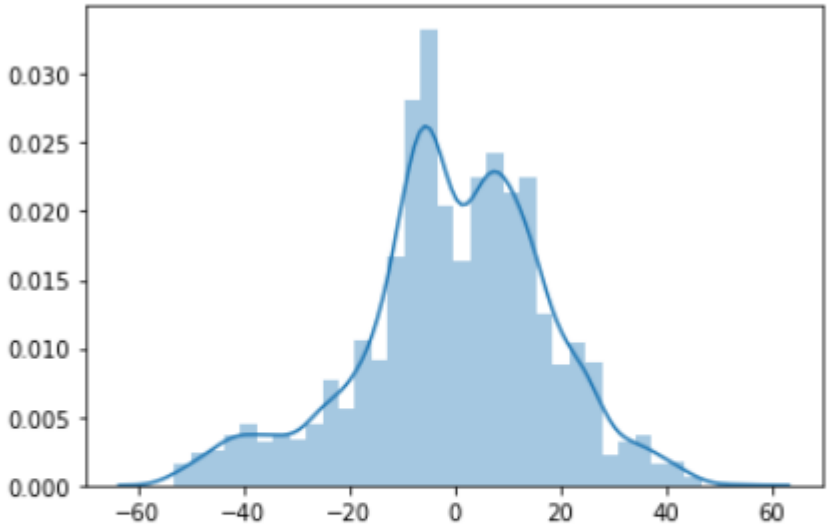

```
MAE: 11.119955248920911
MSE: 203.82449147580922
RMSE: 14.276711507760085
```

In this plot we can observe that most of our values are 0 or close to 0. Therefore we can state that the Lasso regression model works fine.

## Random Forest Model

Evaluating the Random Forest Regression model using Distplot and Sklearn Metrics:

```
print(sns.distplot(y_test-prediction))
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```




```
MAE: 13.915122417126046
MSE: 318.44189128913825
RMSE: 17.844940215342227
```

In this plot we can observe that not many of our values are 0 or close to 0 when compared to Lasso Regression. Also the error values are higher than Lasso Regression. Therefore we can state that the Lasso regression model has a better accuracy than Random Forest.

## Saving model for deployment

By analyzing the above models we can conclude that Lasso Regression works better on our dataset as it had lower error metric values than Random Forest Regressor. Now we will save the model in a pickle file.

```
# Creating a pickle file for the classifier
filename = 'Batting-score-LassoReg-model.pkl'
pickle.dump(lasso_regressor, open(filename, 'wb'))
```

## Step 7: Deploy the model

For deployment we will use flask framework. I have made a python file known as ‘app.py’. What this code does is, it will give us access to the ‘index.html’ and ‘predict.html’ files. We have used the POST method to call.

```
# Importing essential libraries
from flask import Flask, render_template, request
import pickle
import numpy as np

# Load the Random Forest CLassifier model
filename = 'Batting-score-LassoReg-model.pkl'
regressor = pickle.load(open(filename, 'rb'))

app = Flask(__name__)

@app.route('/')
def home():
        return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    temp_array = list()

    if request.method == 'POST':

        batting_team = request.form['batting-team']
        if batting_team == 'Chennai Super Kings':
            temp_array = temp_array + [1,0,0,0,0,0,0,0]
        elif batting_team == 'Delhi Daredevils':
            temp_array = temp_array + [0,1,0,0,0,0,0,0]
```

```
elif batting_team == 'Delhi Daredevils':
    temp_array = temp_array + [0,1,0,0,0,0,0,0]
elif batting_team == 'Kings XI Punjab':
    temp_array = temp_array + [0,0,1,0,0,0,0,0]
elif batting_team == 'Kolkata Knight Riders':
    temp_array = temp_array + [0,0,0,1,0,0,0,0]
elif batting_team == 'Mumbai Indians':
    temp_array = temp_array + [0,0,0,0,1,0,0,0]
elif batting_team == 'Rajasthan Royals':
    temp_array = temp_array + [0,0,0,0,0,1,0,0]
elif batting_team == 'Royal Challengers Bangalore':
    temp_array = temp_array + [0,0,0,0,0,0,1,0]
elif batting_team == 'Sunrisers Hyderabad':
    temp_array = temp_array + [0,0,0,0,0,0,0,1]


bowling_team = request.form['bowling-team']
if bowling_team == 'Chennai Super Kings':
    temp_array = temp_array + [1,0,0,0,0,0,0,0]
elif bowling_team == 'Delhi Daredevils':
    temp_array = temp_array + [0,1,0,0,0,0,0,0]
elif bowling_team == 'Kings XI Punjab':
    temp_array = temp_array + [0,0,1,0,0,0,0,0]
elif bowling_team == 'Kolkata Knight Riders':
    temp_array = temp_array + [0,0,0,1,0,0,0,0]
elif bowling_team == 'Mumbai Indians':
    temp_array = temp_array + [0,0,0,0,1,0,0,0]
elif bowling_team == 'Rajasthan Royals':
```

```
        temp_array = temp_array + [0,0,0,0,0,1,0,0]
    elif bowling_team == 'Royal Challengers Bangalore':
        temp_array = temp_array + [0,0,0,0,0,0,1,0]
    elif bowling_team == 'Sunrisers Hyderabad':
        temp_array = temp_array + [0,0,0,0,0,0,0,1]



    overs = float(request.form['overs'])
    runs = int(request.form['runs'])
    wickets = int(request.form['wickets'])
    runs_in_prev_5 = int(request.form['runs_in_prev_5'])
    wickets_in_prev_5 = int(request.form['wickets_in_prev_5'])

    temp_array = temp_array + [overs, runs, wickets, runs_in_prev_5, wickets_in_prev_5]

    data = np.array([temp_array])
    my_prediction = int(regressor.predict(data)[0])

    return render_template('result.html', lower_limit = my_prediction-10, upper_limit = my_p



__name__ == '__main__':
    app.run(debug=True)
```

When using the flask framework we need to make 2 folders: static and templates. The Html files should be stored in the template folder while the images and css files should be stored in the static folder. Here below is the code for html file of our home page: index.html.

```
<!DOCTYPE
html>
            <html>
            <head>
              <title>IPL Score Predictor</title>
                <link rel="stylesheet" type="text/css" href="{{ url_for('static',
            filename='styles.css') }}">
                <link rel="shortcut icon" href="{{ url_for('static', filename='ipl-favicon.ico')
            }}">

                <script src="https://kit.fontawesome.com/f13b0bc903.js"
            crossorigin="anonymous"></script>
                </head>
              <body>
                  <h1>IPL Batting Score Predictor 🏆</h1>
                  <div class = "register">
                  <form id="register" action="{{ url_for('predict') }}"  method="post">

                      <label> Batting Team: </label><br>
                      <select name="batting-team" style="font-size:10pt;">
                                  <option value="none">--- Select a Batting team ---</option>
                                  <option value="Mumbai Indians">Mumbai Indians</option>
                                  <option value="Kolkata Knight Riders">Kolkata Knight
            Riders</option>
                                  <option value="Chennai Super Kings">Chennai Super
            Kings</option>
                                  <option value="Rajasthan Royals">Rajasthan Royals</option>
                                  <option value="Kings XI Punjab">Kings XI Punjab</option>
                                  <option value="Royal Challengers Bangalore">Royal Challengers
            Bangalore</option>
                                  <option value="Delhi Daredevils">Delhi Daredevils</option>
                                  <option value="Sunrisers Hyderabad">Sunrisers
            Hyderabad</option>
                              </select><br><br>
                      <label> Bowling Team: </label><br>
                      <select  name="bowling-team" style="font-size:10pt;">
                                  <option value="none">--- Select a Bowling team ---</option>
                                  <option value="Mumbai Indians">Mumbai Indians</option>
```

```
                        <option value="Kolkata Knight Riders">Kolkata Knight
Riders</option>
                        <option value="Chennai Super Kings">Chennai Super
Kings</option>
                        <option value="Rajasthan Royals">Rajasthan Royals</option>
                        <option value="Kings XI Punjab">Kings XI Punjab</option>
                        <option value="Royal Challengers Bangalore">Royal Challengers
Bangalore</option>
                        <option value="Delhi Daredevils">Delhi Daredevils</option>
                        <option value="Sunrisers Hyderabad">Sunrisers
Hyderabad</option>
                    </select><br><br>
          <label> Venue: </label><br>
          <select name="venue" style="font-size:10pt;">
                        <option value="none">--- Select the Stadium ---</option>
                        <option value="Eden Gardens">Eden Gardens</option>
                        <option value="Feroz Shah Kotla">Feroz Shah Kotla</option>
                        <option value="M Chinnaswamy Stadium">M Chinnaswamy
Stadium</option>
                        <option value="MA Chidambaram Stadium, Chepauk">MA Chidambaram
Stadium</option>
                        <option value="Sawai Mansingh Stadium">Sawai Mansingh
Stadium</option>
                        <option value="Punjab Cricket Association Stadium,
Mohali">Punjab Cricket Association Stadium</option>
                        <option value="Rajiv Gandhi International Stadium,
Uppal">Rajiv Gandhi International Stadium</option>
                        <option value="Wankhede Stadium">Wankhede Stadium</option>
                    </select><br><br>

          <label> Overs Bowled: </label><br>
          <input class="form-input" type="text" name="overs" placeholder="Overs (>=
5.0) Eg. 7.2"><br><br>
          <label> Runs Scored: </label><br>
          <input class="form-input" type="text" name="runs" placeholder="Eg.
64"><br><br>
          <label> Wickets Fallen: </label><br>
          <input class="form-input" type="text" name="wickets" placeholder="Eg.
4"><br><br>
          <label> Runs Scored in previous 5 Overs: </label><br>
          <input class="form-input" type="text" name="runs_in_prev_5" placeholder="Eg.
42"><br><br>
          <label> Wickets taken in previous 5 Overs: </label><br>
          <input class="form-input" type="text" name="wickets_in_prev_5"
```

```
placeholder="Eg. 3"><br><br>

          <input type="submit" class="myButton" value="Predict Score">
```

Further we also created a prediction webpage for displaying the result. Here below is the code for html file of that webpage: result.html

```
<!DOCTYPE
html>
            <html>
            <head>
              <title>IPL Score Predictor</title>
                <link rel="stylesheet" type="text/css" href="{{ url_for('static',
            filename='styles.css') }}">
                <link rel="shortcut icon" href="{{ url_for('static', filename='ipl-
            favicon.ico') }}">

                <script src="https://kit.fontawesome.com/f13b0bc903.js"
            crossorigin="anonymous"></script>
                </head>

                <body>
                    <h1>  IPL Batting Score Predictor 🏆</h1>

                                <!-- Result -->
                                <div class="register" >
                            <center>
                            <p style="font-size:30px;">The final predicted score
            (range):<br> {{ lower_limit }} to {{ upper_limit }}</p><br><br>
                            </center>

                                </div>
                    <br><br><br>
                          <center>
                                <p style="font-size:20px;">Made by Atharva
            Patil.</p><br>

                            <a href="https://github.com/Atharva1604"><i class="fab
            fa-github fa-lg contact-icon"></i></a>  
                           <a href="https://www.linkedin.com/in/atharva-patil-
            a79a84176/"><i class="fab fa-linkedin fa-lg contact-
```

```
icon"></i></a><br><br>
            </center>
        <center>
        <carousel>
      <img src="{{ url_for('static', filename= 'mi1.jpg') }}">
      <img src="{{ url_for('static', filename= 'srh1.png') }}">
      <img src="{{ url_for('static', filename= 'rcb1.png') }}">
      <img src="{{ url_for('static', filename= 'dc1.png') }}">
      <img src="{{ url_for('static', filename= 'rr1.png') }}">
      <img src="{{ url_for('static', filename= 'kkr1.jpg') }}">
      <img src="{{ url_for('static', filename= 'kxip1.png') }}">
      <img src="{{ url_for('static', filename= 'csk1.png') }}">
    </carousel>


        </center>


        </body>
    </html>
```

For styling and design we have to use CSS file and store it in static folder. Here is the code for that:

```
*{
        margin: 0;
        padding: 0;
    }

    h1{
        text-align: center;
        padding: 20px;
    }
    h2{
        text-align: center;
        padding: 20px;
    }
    .inside {
        position: absolute;
        top: 0;
        right: 0;
        bottom: 0;
        left: 0;
        display: flex;
```

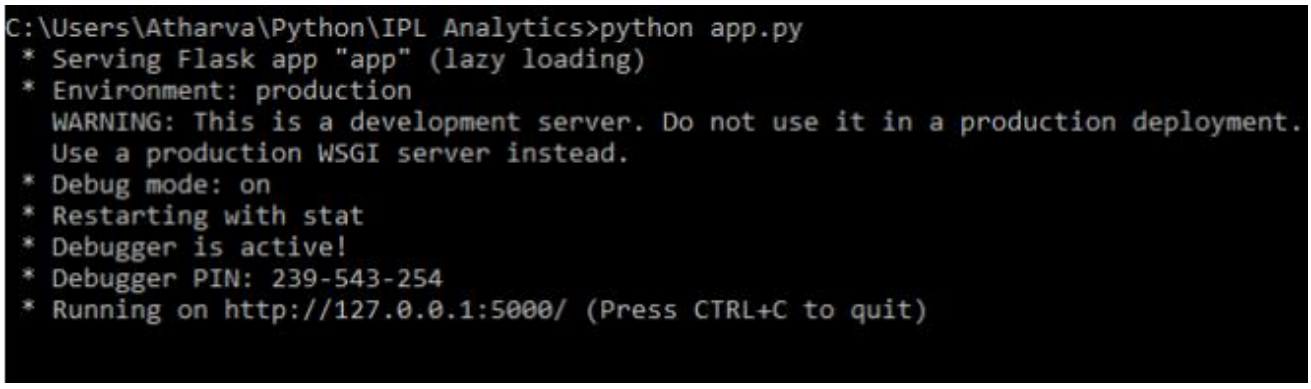
```
    justify-content: center;
    align-items: center;
}
.wrap {
    height: 500px;
    width: 500px;
    position: relative;
}

.register
{

        background-color:  #00237d;
        width: 500px;
        margin: 0px 0px 0px 450px;
        color: white;
        font-size: 18px;
        padding: 20px;
        border-radius: 10px;
}
#register{
    margin-left: 100px;
}
label{
    color: white;
    font-family: sans-serif;
    font-size: 18px;
    font-style: normal;
}
#name{
    width: 300px;
    border: none;
    border-radius: 3px;
    outline: 0px;
    padding: 7px;
}
#ph{
    width: 65px;
    padding: 7px;
    border: none;
    border-radius: 3px;
    outline: 0;
```

```
}
#num{
    width: 230px;
    padding: 7px;
    border: none;
    border-radius: 3px;
    outline: 0;
}
#sub{
    width: 200px;
    padding: 7px;
    font-size: 12px;
    font-family: sans-serif;
    font-style: normal;
    border: none;
    border-radius: 3px;
    outline: 0;
}
.myButton {
        box-shadow:inset 0px 1px 0px 0px #fbafe3;
        background:linear-gradient(to bottom, #ff5bb0 5%, #ef027d 100%);
        background-color:#ff5bb0;
        border-radius:6px;
        border:1px solid #ee1eb5;
        display:inline-block;
        cursor:pointer;
        color:#ffffff;
        font-family:Arial;
        font-size:15px;
        font-weight:bold;
        padding:6px 24px;
        text-decoration:none;
        text-shadow:0px 1px 0px #c70067;
}
.myButton:hover {
        background:linear-gradient(to bottom, #ef027d 5%, #ff5bb0 100%);
        background-color:#ef027d;
}
.myButton:active {
        position:relative;
        top:1px;
}
```

```
/*animations*/
carousel {
  max-width: 100%;
  height: 250px;
  max-height: 250px;
  margin-bottom: 20px;
  position: relative;
  overflow: hidden;
  display: block;

}
carousel img {
  max-width: 100%;
  width: 600px;
  height: auto;
  max-height: 250px;
  position: absolute;
  animation: slide 40s infinite;


}
carousel img:nth-child(2) {
margin-left: 50%;
}
carousel img:nth-child(3) {
margin-left: 100%;
}
carousel img:nth-child(4) {
margin-left: 150%;
}
carousel img:nth-child(5) {
margin-left: 200%;
}
carousel img:nth-child(6) {
margin-left: 250%;
}
carousel img:nth-child(7) {
margin-left: 300%;
}
carousel img:nth-child(8) {
margin-left: 350%;
```

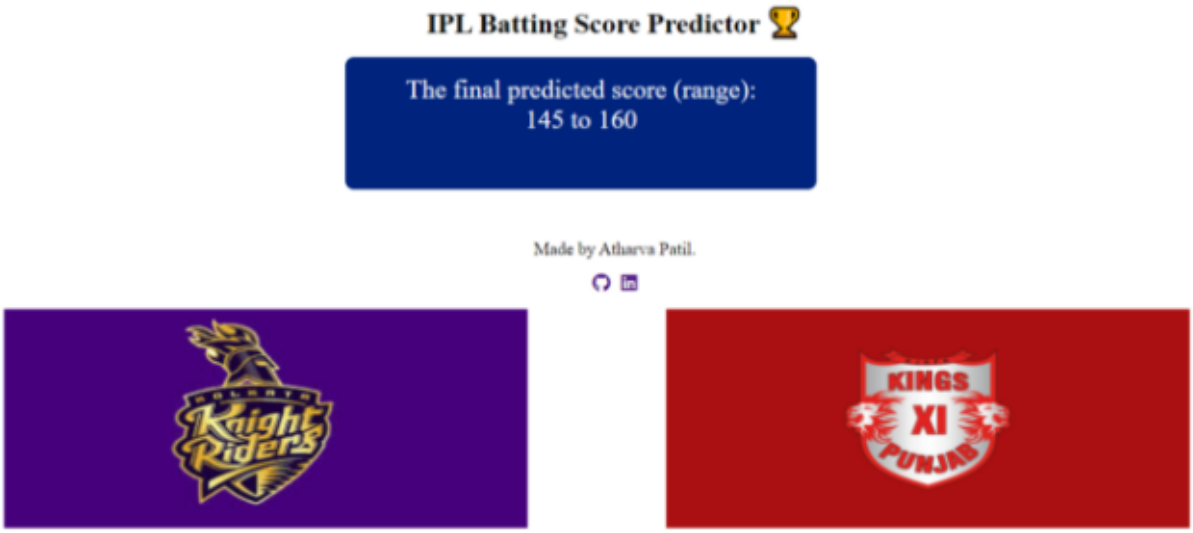
```
}

@keyframes slide {
  0% {
    left: 0;
  }
  50% {
    left: -300%;
  }
  100% {
    left: 0;
  }
}
```

Now we are ready to run our model on our local machine. Open the command promt and first change directory to the folder where we have the project saved. Then run python app.py.



Now on your browser open and run the application. Below are the images of the User Interface.


Home Page




Prediction page

**THANK YOU**