IPL score prediction system using Machine Learning

In [125…
```python
# Importing Necessary Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [126…
```python
#Importing dataset
ipl_df = pd.read_csv("D:\\data_analysis\\Ml projects\\IPl_win_predictor\\New folder\\ipl_data.csv")
```

Exploratory Data Analysis:-

In [127…
```python
# First 5 Columns Data
ipl_df.head()
```

Out[127]:

	mid	date	venue	bat_team	bowl_team	batsman	bowler	runs	wickets	overs	runs_last_5	wickets_last_5	striker	non-striker	tot
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	22
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	22
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	22
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	22
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	22

In [128…
```python
# Describing the ipl_dfset
ipl_df.describe()
```

Out[128]:

	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

In [129…
```python
# Information about Each Column
ipl_df.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   bat_team        76014 non-null  object
 4   bowl_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 [130…

```
# Number of Unique Values in each column
ipl_df.nunique()
```

Out[130]:

```
mid               617
date              442
venue              35
bat_team           14
bowl_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
```

In [131…

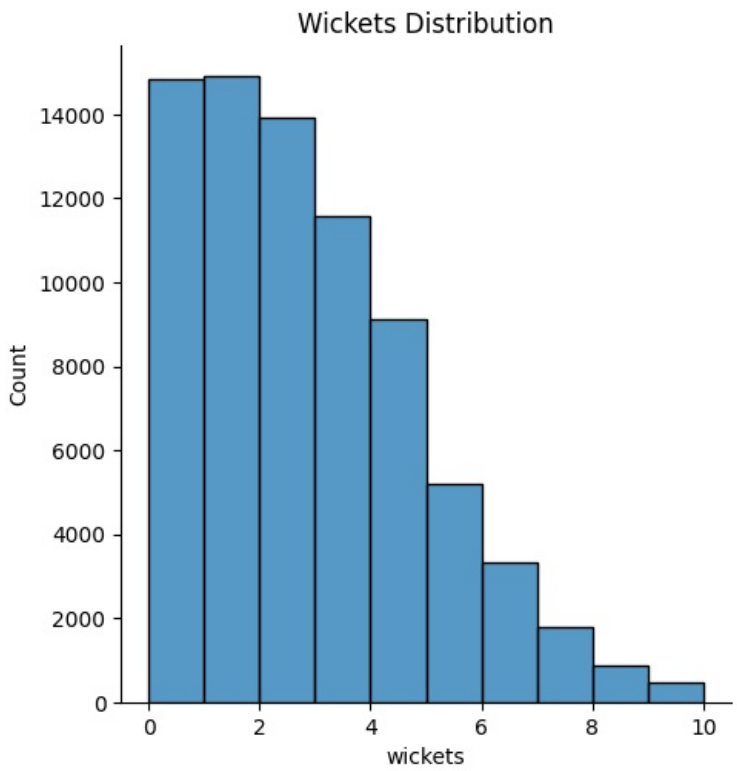
```
# ipl_df types of all Columns
ipl_df.dtypes
```

Out[131]:

```
mid                 int64
date               object
venue              object
bat_team           object
bowl_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
```

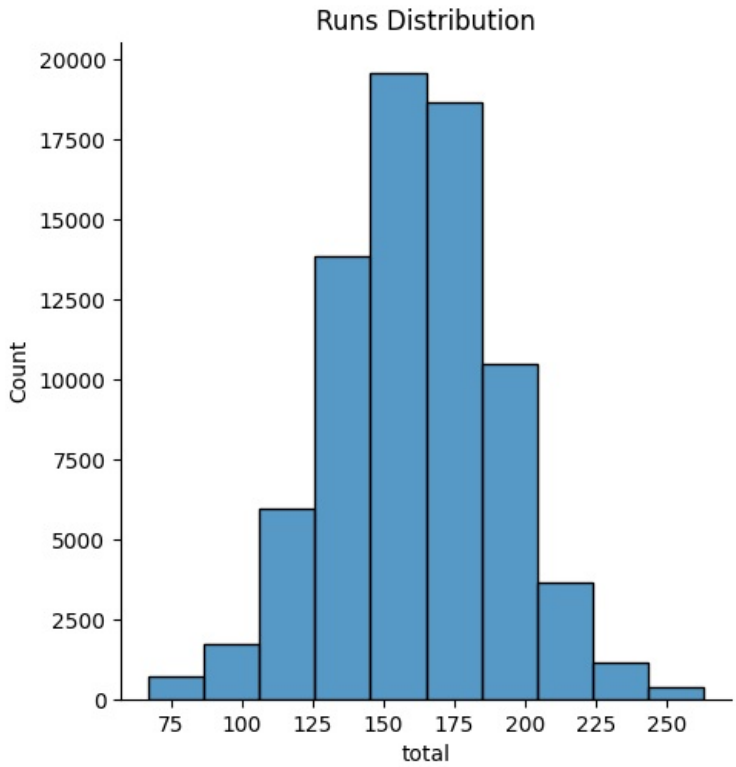
In [132…

```
#Wickets Distribution
sns.displot(ipl_df['wickets'],kde=False,bins=10)
plt.title("Wickets Distribution")

plt.show()
```



In [133…

```
#Runs Distribution
sns.displot(ipl_df['total'],kde=False,bins=10)
plt.title("Runs Distribution")

plt.show()
```



Data Cleaning:-

Removing Irrelevant Data columns

In [134…
```
# Names of all columns
ipl_df.columns
```

Out[134]:
```
Index(['mid', 'date', 'venue', 'bat_team', 'bowl_team', 'batsman', 'bowler',
       'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5', 'striker',
       'non-striker', 'total'],
      dtype='object')
```

In [135…
```
irrelevant = ['mid', 'date', 'venue','batsman', 'bowler', 'striker', 'non-striker']
print(f'Before Removing Irrelevant Columns : {ipl_df.shape}')
ipl_df = ipl_df.drop(irrelevant, axis=1) # Drop Irrelevant Columns
print(f'After Removing Irrelevant Columns : {ipl_df.shape}')
ipl_df.head()
```

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

Out[135]:

	bat_team	bowl_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

In [136…
```
# 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 [137…
```
print(f'Before Removing Inconsistent Teams : {ipl_df.shape}')
ipl_df = ipl_df[(ipl_df['bat_team'].isin(const_teams)) & (ipl_df['bowl_team'].isin(const_teams))]
print(f'After Removing Irrelevant Columns : {ipl_df.shape}')
print(f"Consistent Teams : \n{ipl_df['bat_team'].unique()}")
ipl_df.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[137]:

	bat_team	bowl_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

In [138…

```
print(f'Before Removing Overs : {ipl_df.shape}')
ipl_df = ipl_df[ipl_df['overs'] >= 5.0]
print(f'After Removing Overs : {ipl_df.shape}')
ipl_df.head()
```

```
Before Removing Overs : (53811, 8)
After Removing Overs : (40108, 8)
```

Out[138]:

	bat_team	bowl_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
32	Kolkata Knight Riders	Royal Challengers Bangalore	61	0	5.1	59	0	222
33	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.2	59	1	222
34	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.3	59	1	222
35	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.4	59	1	222
36	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.5	58	1	222

Data Preprocessing and Encoding

In [139…

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
for col in ['bat_team', 'bowl_team']:
  ipl_df[col] = le.fit_transform(ipl_df[col])
ipl_df.head()
```

Out[139]:

	bat_team	bowl_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 Encoding and Column Transformation

In [140…

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

In [141…

```
ipl_df = np.array(columnTransformer.fit_transform(ipl_df))
```

In [142…

```
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 Roya
              'batting_team_Royal Challengers Bangalore', 'batting_team_Sunrisers Hyderabad',
              'bowling_team_Chennai Super Kings', 'bowling_team_Delhi Daredevils', 'bowling_team_Kings XI Punja
              'bowling_team_Kolkata Knight Riders', 'bowling_team_Mumbai Indians', 'bowling_team_Rajasthan Roya
              'bowling_team_Royal Challengers Bangalore', 'bowling_team_Sunrisers Hyderabad', 'runs', 'wickets'
       'runs_last_5', 'wickets_last_5', 'total']
df = pd.DataFrame(ipl_df, columns=cols)
```

In [143…

```
# Encoded Data
df.head()
```

Out[143]:

	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	battir
0	0.0	0.0	0.0	1.0	0.0	0.0	
1	0.0	0.0	0.0	1.0	0.0	0.0	
2	0.0	0.0	0.0	1.0	0.0	0.0	
3	0.0	0.0	0.0	1.0	0.0	0.0	
4	0.0	0.0	0.0	1.0	0.0	0.0	

5 rows × 22 columns

Model Building

Prepare Train and Test Data

In [144…

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

In [145…

```
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, s
print(f"Training Set : {train_features.shape}\nTesting Set : {test_features.shape}")
```

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

Machine Learning Algorithms

In [146…

```
models = dict()
```

1. Decision Tree algorithm

In [147…

```
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor()
# Train Model
tree.fit(train_features, train_labels)
```

Out[147]:

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [148…

```
# 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 : 99.99%
Test Score : 86.55%
```

In [149…

```
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): 3.9523186237845924
Mean Squared Error (MSE): 120.64207803540265
Root Mean Squared Error (RMSE): 10.983718770771702
```

2. Linear Regression algorithm

In [150…

```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
# Train Model
linreg.fit(train_features, train_labels)
```

Out[150]:

```
▾ LinearRegression
LinearRegression()
```

In [151…

```
# 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 : 65.72%
Test Score : 66.66%
```

In [152…

```
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): 12.965220034972608
Mean Squared Error (MSE): 299.06164448372425
Root Mean Squared Error (RMSE): 17.293398870196807
```

3.Random Forest algorithm

In [153…

```
from sklearn.ensemble import RandomForestRegressor
forest = RandomForestRegressor()
# Train Model
forest.fit(train_features, train_labels)
```

Out[153]:

```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [154…

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

```
Train Score : 99.04%
Test Score : 93.55%
```

In [155…

```
print("---- Random Forest Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(test_labels, forest.predict(test_features))))
print("Mean Squared Error (MSE): {}".format(mse(test_labels, forest.predict(test_features))))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(test_labels, forest.predict(test_features)))))
```

```
---- Random Forest Regression - Model Evaluation ----
Mean Absolute Error (MAE): 4.475842135516219
Mean Squared Error (MSE): 57.81990334626344
Root Mean Squared Error (RMSE): 7.6039399883391665
```

4.Support Vector Machine

In [156…

```
from sklearn.svm import SVR
svm = SVR()
# Train Model
svm.fit(train_features, train_labels)
```

Out[156]:

```
▾ SVR
SVR()
```

In [157…

```
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 : 57.16%
Test Score : 58.67%
```

In [158…

```
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): 14.550524392848825
Mean Squared Error (MSE): 370.7817563398098
Root Mean Squared Error (RMSE): 19.255694127707
```

5.K-Nearest neighbour algorithm

In [159…

```
from sklearn.neighbors import KNeighborsRegressor
knr = KNeighborsRegressor()
# Train Model
knr.fit(train_features, train_labels)
```

Out[159]:

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

In [160…

```
train_score_knr = str(knr.score(train_features, train_labels)*100)
test_score_knr = str(knr.score(test_features, test_labels)*100)
print(f'Train Score : {train_score_knr[:5]}%\nTest Score : {test_score_knr[:5]}%')
models["knr"] = test_score_knr
```

```
Train Score : 86.61%
Test Score : 78.26%
```

In [161…

```
print("---- KNR - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(test_labels, knr.predict(test_features))))
print("Mean Squared Error (MSE): {}".format(mse(test_labels, knr.predict(test_features))))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(test_labels, knr.predict(test_features)))))
```

```
---- KNR - Model Evaluation ----
Mean Absolute Error (MAE): 9.706332585390179
Mean Squared Error (MSE): 195.00653702318624
Root Mean Squared Error (RMSE): 13.96447410478412
```

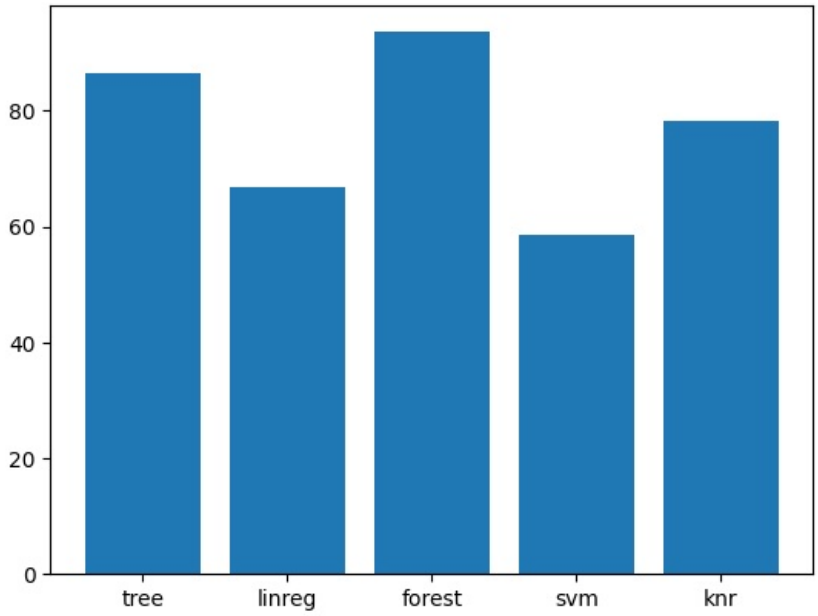
Best Model

In [162…

```
import matplotlib.pyplot as plt
model_names = list(models.keys())
accuracy = list(map(float, models.values()))
# creating the bar plot
plt.bar(model_names, accuracy)
```

Out[162]:

```
<BarContainer object of 5 artists>
```



From above, we can see that Random Forest performed the best, closely followed by Decision Tree and KNR. So we will be choosing Random Forest for the final model

Predictions

In [163…

```
def score_predict(batting_team, bowling_team, runs, wickets, overs, runs_last_5, wickets_last_5, model=forest):
  prediction_array = []
  # Batting Team
  if batting_team == 'Chennai Super Kings':
    prediction_array = prediction_array + [1,0,0,0,0,0,0,0]
  elif batting_team == 'Delhi Daredevils':
    prediction_array = prediction_array + [0,1,0,0,0,0,0,0]
  elif batting_team == 'Kings XI Punjab':
    prediction_array = prediction_array + [0,0,1,0,0,0,0,0]
  elif batting_team == 'Kolkata Knight Riders':
    prediction_array = prediction_array + [0,0,0,1,0,0,0,0]
  elif batting_team == 'Mumbai Indians':
    prediction_array = prediction_array + [0,0,0,0,1,0,0,0]
  elif batting_team == 'Rajasthan Royals':
    prediction_array = prediction_array + [0,0,0,0,0,1,0,0]
  elif batting_team == 'Royal Challengers Bangalore':
    prediction_array = prediction_array + [0,0,0,0,0,0,1,0]
  elif batting_team == 'Sunrisers Hyderabad':
    prediction_array = prediction_array + [0,0,0,0,0,0,0,1]
```

```
# Bowling Team
if bowling_team == 'Chennai Super Kings':
  prediction_array = prediction_array + [1,0,0,0,0,0,0,0]
elif bowling_team == 'Delhi Daredevils':
  prediction_array = prediction_array + [0,1,0,0,0,0,0,0]
elif bowling_team == 'Kings XI Punjab':
  prediction_array = prediction_array + [0,0,1,0,0,0,0,0]
elif bowling_team == 'Kolkata Knight Riders':
  prediction_array = prediction_array + [0,0,0,1,0,0,0,0]
elif bowling_team == 'Mumbai Indians':
  prediction_array = prediction_array + [0,0,0,0,1,0,0,0]
elif bowling_team == 'Rajasthan Royals':
  prediction_array = prediction_array + [0,0,0,0,0,1,0,0]
elif bowling_team == 'Royal Challengers Bangalore':
  prediction_array = prediction_array + [0,0,0,0,0,0,1,0]
elif bowling_team == 'Sunrisers Hyderabad':
  prediction_array = prediction_array + [0,0,0,0,0,0,0,1]
prediction_array = prediction_array + [runs, wickets, overs, runs_last_5, wickets_last_5]
prediction_array = np.array([prediction_array])
pred = model.predict(prediction_array)
return int(round(pred[0]))
```

Sample:

```
Batting Team : Delhi Daredevils
Bowling Team : Chennai Super Kings
Final Score : 147/9
```

In [164…

```
batting_team='Delhi Daredevils'
bowling_team='Chennai Super Kings'
score = score_predict(batting_team, bowling_team, overs=10.2, runs=68, wickets=3, runs_last_5=29, wickets_last_
print(f'Predicted Score : {score} || Actual Score : 147')
```

```
Predicted Score : 147 || Actual Score : 147
```

```
C:\Users\91982\anaconda3\lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature name
s, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: