

End-to-end machine learning project to predict T20 score.

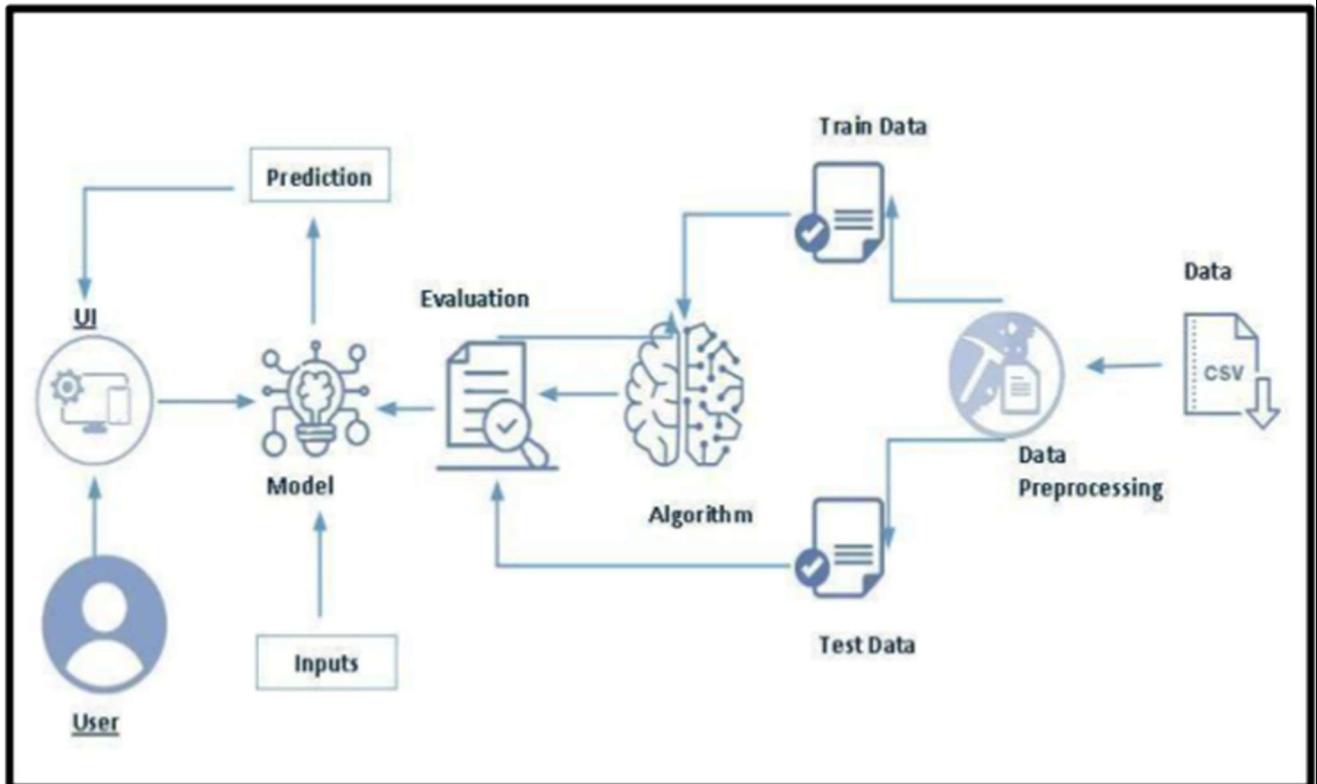
The objective of this project is to develop an end-to-end machine learning solution for predicting the t20 score of the batting team. The proposed solution involves the use of Machine learning algorithms to extract relevant features from the input and predict the accurate score.

Creating an end-to-end machine learning project to predict T20 cricket scores can offer a range of benefits, both from a sports analytics perspective and as a showcase of machine learning capabilities. Here are some of the key advantages:

1. **Insightful Analytics:** Developing a T20 score prediction model can provide deep insights into the factors that influence team performance and scoring in cricket matches. This could include player statistics, pitch conditions, team composition, weather conditions, and more.
2. **Strategic Decision Making:** Cricket teams, coaches, and analysts can use the predictive model to make informed decisions during matches. This could involve adjusting strategies based on predicted scores, understanding the impact of different factors on the outcome, and optimizing team compositions.
3. **Engaging Fan Experience:** Fans of the sport can benefit from more engaging and interactive experiences. Predicted scores can be displayed in real-time during live broadcasts, enhancing fan engagement and offering viewers a better understanding of the ongoing match dynamics.
4. **Betting and Fantasy Sports:** Predictive models are often sought after by individuals engaged in sports betting and fantasy sports leagues. Accurate score predictions can be used to make informed betting decisions or create more competitive fantasy teams.

Let us look at the Technical Architecture of the project.

Technical Architecture:



Project Flow:

- The user interacts with the UI to enter the data.
 - Uploaded input is analyzed by the model which is integrated/developed by you.
 - Once the model analyzes the input the prediction is showcased on the UI. To accomplish this, we have to complete all the activities listed below,
- Define Problem / Problem Understanding
 - Specify the business problem
 - Business requirements
 - Literature Survey.
 - Data Collection & Preparation
 - Collect the dataset
 - Data Preparation
 - Exploratory Data Analysis
 - Descriptive statistical
 - Visual Analysis
 - Model Building
 - Building a model.
 - Training the model.
 - Testing the model
 - Model Deployment
 - Save the best model
 - Integrate with Web Framework

Prior Knowledge:

To complete this project, you must require following software's , concepts and packages

- VS Code : o Refer to the link below to download VS Code. o Link : <https://code.visualstudio.com/download> • Create Project: o Open VS Code. o Create a Folder and name it "T20_Score_Predictor" .
- Machine Learning Concepts
 - o Linear Regression: <https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86> o XGBRegressor : <https://medium.com/@aryanbajaj13/prediction-using-xgb-regressor-using-python-3-ad1a57e105f>
 - o Random Forest : <https://medium.com/towards-data-science/understanding-random-forest-58381e0602d2>
- Web concepts: Get the gist on streamlit : <https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/>

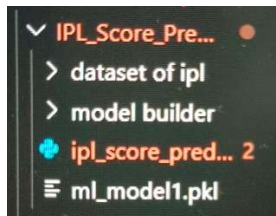
Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques of Machine Learning.
- Know how to pre-process/clean the data using different data preprocessing techniques.
- Know how to build a web application using the Stream lit framework.

Project Structure:

Create the Project folder which contains files as shown below



- We are building a Streamlit application which needs a python script web.py for a website.
- Model folder contains your saved models. And your code for building/training/testing the model.
- Dataset folder contains your data.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem.

In the profession of cricket score prediction, a variety of strategies are used to forecast the innings score of a cricket match. Numerous systems and prediction computations are used to forecast the outcomes. The problem is addressed that the inaccurate predictions and analysis by these systems which may leads to loose of the game by wrong strategy or wrong calculation. The models are not user-friendly interface not real time models.

Activity 2: Business requirements

Here are some potential business requirements for a t20 score predictor using machine learning:

1. **Research and Analysis:** Building the prediction model involves thorough analysis of historical match data, player statistics, pitch conditions, and other relevant factors. This research can contribute to a better understanding of the game and its nuances.
2. **Machine Learning Skill Showcase:** Developing an end-to-end machine learning project demonstrates your skills as a data scientist or machine learning engineer. This kind of project showcases your ability to gather, preprocess, and analyze data, select appropriate features, choose a suitable model, train and fine-tune it, and deploy it to make predictions.
3. **Learning Opportunity:** Creating such a project allows you to dive into various aspects of machine learning and data science, from data preprocessing and feature engineering to model selection and evaluation. It's a great way to learn about different algorithms, techniques, and tools.

Activity 3: Literature Survey

We studied papers based on the area of our research i.e. Cricket prediction. We studied 8 IEEE papers and drew some conclusions from the study. A comparison of all 8 IEEE papers has been made [1]. The work proposed in [2] deals with the score prediction of the first innings and also predicts the outcome of the match after the second innings. Linear Regression algorithm is used to predict the first innings score and outcome prediction is done by using Naive Bayes Classifier. In [3], the research aims at predicting the result of an ongoing cricket match on an over-by-over basis based on the information and data that is available from each over. The author tests the datasets on various machine learning models. It has been found that the Random Forest algorithm has the highest accuracy. The work proposed in [4] deals with the sentiments. The author predicts the outcome and man of the match by using twitter-based positive and negative sentiment analysis. Naive Bayes classifier, SVM, Random Forest algorithm, and Logistic Regression, are some of the models used for prediction. In [5], cricket squad analysis is done. This paper provides a mathematical approach to select the players. RMSE value of Multiple Random Forest Regression is greater than LR, SVR, and Decision Tree. The work proposed in [6], deals with the player's performance prediction in ODI matches. This paper proposes the model which consists of statistical data of Bangladesh players. The main aim of this research is to predict the performance of players based on the records using SVM with linear kernel and SVM with the polynomial kernel. In [7], CNN and Feature encoding is used to predict the outcome of cricket matches. The research predicts the outcome of a cricket match is predicted even before the match starts. The accuracy of shallow CNN is over 70%. The work proposed in [8] is the research of various papers. In this paper author analyses the work done by various authors in the cricket prediction domain. In [9], outcome prediction of ODI matches is done using decision trees and MLP networks. A comparative study is done between MLP and Decision trees and final results are given. In this study, a comparative analysis of the predictions generated by 2 different supervised classification models was performed for the same input dataset.

Activity 4: Social or Business Impact.

1. **Improved Decision Making:** Cricket teams can use accurate score predictions to optimize in-game strategies, player selection, and overall team performance, potentially leading to more victories and a stronger fan following.
2. **Sponsorship and Advertising:** Enhanced engagement and accurate predictions can attract more sponsors and advertisers to cricket events. Companies may be eager to associate their brand with a sport that engages a tech-savvy audience.
3. **Media and Broadcast Enhancements:** Media outlets can integrate predicted scores into their live broadcasts and coverage, providing real-time insights to viewers. This can make broadcasts more engaging and informative.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset.

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Download only "Ipl_dataset csv" data.

Make sure your directory looks like below

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Under the Training folder create a Training.ipynb file in jupyter and start writing code in this file.

Activity 2: Importing the libraries

Import the necessary libraries as shown in the image.

```
# Importing Necessary Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Activity 1.2: Importing the dataset and analyse the data

Load the dataset

```
: #Importing dataset
ipl_df = pd.read_csv(r"C:\Users\JAYANTH VARMA\Downloads\ipl_data.csv")
print(f"Dataset successfully Imported of Shape : {ipl_df.shape}")
```

Dataset successfully Imported of Shape : (76014, 15)

```
: # First 5 Columns Data
ipl_df.head()
```

	mid	date	venue	bat_team	bowl_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

```
# Describing the ipl_dfset
ipl_df.describe()
```

	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

Milestone 3: Exploratory Data Analysis

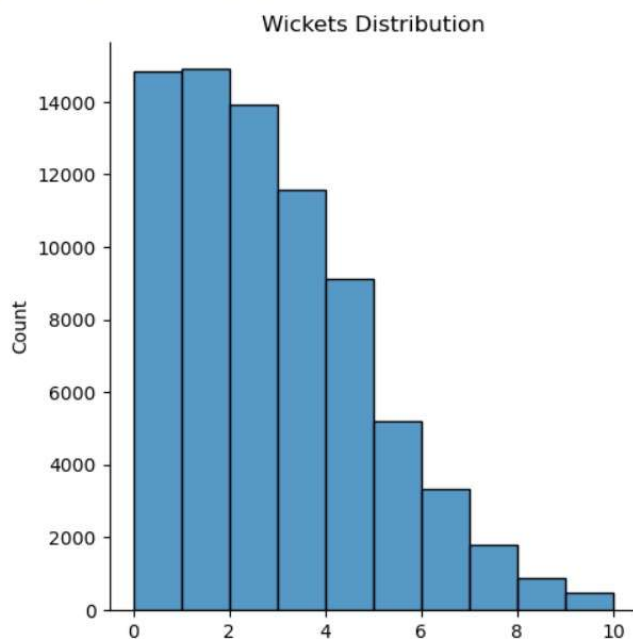
Activity-1: Preprocessing the data

```
# 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
```

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

C:\Users\JAYANTH VARMA\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning
self.figure.tight_layout(*args, **kwargs)



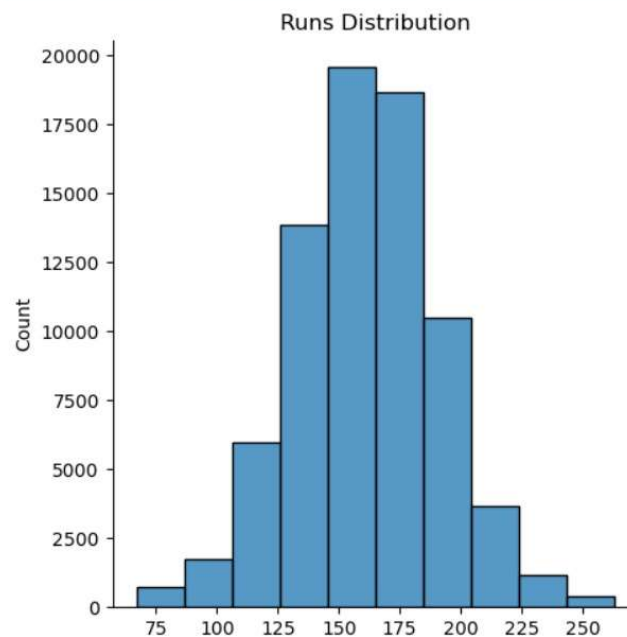
Activity-2: Feature Extraction

```
# 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
```

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

```
C:\Users\JAYANTH VARMA\anaconda3\Lib\site-packages\seaborn\axisgrid.py:  
self.figure.tight_layout(*args, **kwargs)
```



Keeping only Consistent Teams

```
# 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']

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']
```

The above mentioned columns are unnecessary for our prediction. So, let's drop them.

Performing Label Encoding

```
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()
```

	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

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

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

Save the Numpy Array in a new DataFrame with transformed columns

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(ipl_df, columns=cols)

# Encoded Data
df.head()
```

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

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

5 rows × 22 columns

Activity 3: Splitting data into train and test sets

Now let's split the Dataset into train and test sets.

The split will be in 8:2 ratio - train : test respectively.

Prepare Train and Test Data

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

```
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)
```

Milestone 4: Model Training

Activity 1: Model Evolution:

1. Decision Tree Regressor

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

```
: ▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
: # 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 : 84.44%
```

```
: 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): 4.137309897781102
Mean Squared Error (MSE): 137.6454437796061
Root Mean Squared Error (RMSE): 11.732239504016533
```

Linear Regression

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

▼ LinearRegression

```
LinearRegression()
```

```
# 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 : 66.19%
Test Score : 64.82%

```
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): 13.231253750636858
Mean Squared Error (MSE): 311.2042685497275
Root Mean Squared Error (RMSE): 17.64098264127391

Support Vector Machine

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

▼ SVR

```
SVR()
```

```
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 : 57.72%
Test Score : 56.68%

```
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): 14.763365662866732
Mean Squared Error (MSE): 383.20455178342854
Root Mean Squared Error (RMSE): 19.57561114712459

XGBoost

```
: !pip install xgboost
from xgboost import XGBRegressor
xgb = XGBRegressor()
# Train Model
xgb.fit(train_features, train_labels)
```

```
----- 99.7/99.8 MB 671.5 kB/s eta 0:00:01
----- 99.7/99.8 MB 671.5 kB/s eta 0:00:01
----- 99.8/99.8 MB 612.6 kB/s eta 0:00:00
```

Installing collected packages: xgboost
Successfully installed xgboost-2.0.2

```
: XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
```

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

Train Score : 88.95%
Test Score : 85.00%

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

---- XGB Regression - Model Evaluation ----
Mean Absolute Error (MAE): 8.306245825707363
Mean Squared Error (MSE): 132.67378560733977
Root Mean Squared Error (RMSE): 11.518410724025244

KNR

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

```
▼ KNeighborsRegressor
KNeighborsRegressor()
```

```
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 : 86.74%
Test Score : 77.02%

```
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)))))
```

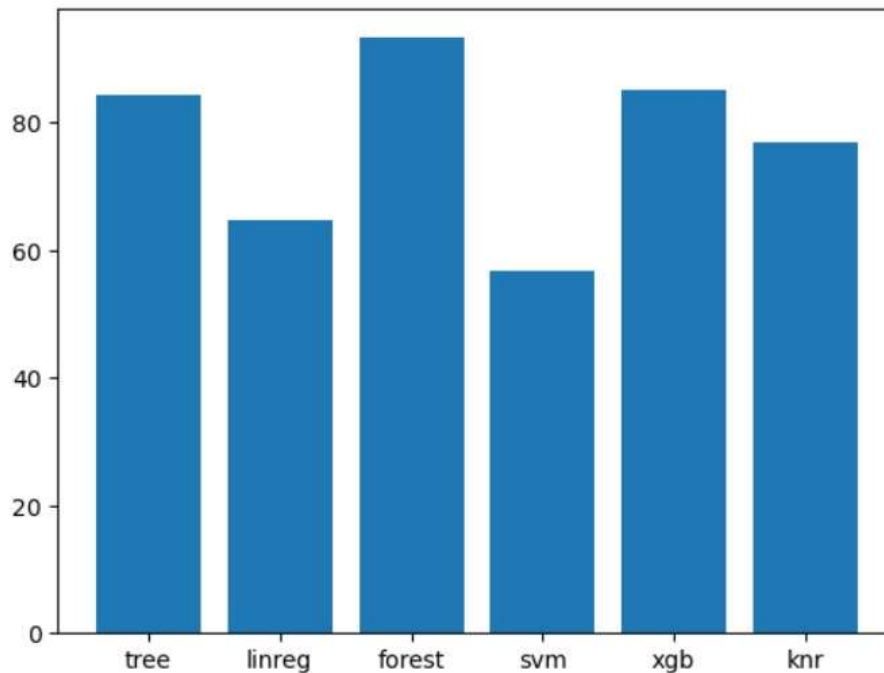
---- KNR - Model Evaluation ----
Mean Absolute Error (MAE): 9.900349040139616
Mean Squared Error (MSE): 203.2539267015707
Root Mean Squared Error (RMSE): 14.256715144154725

Activity 2: Best Model Evolution:

Best Model

```
In [45]: 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[45]: <BarContainer object of 6 artists>
```



```
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 + [1,0,0,0,0,0,0,0,0]
    elif batting_team == 'Delhi Daredevils':
        prediction_array = prediction_array + [0,1,0,0,0,0,0,0,0]
    elif batting_team == 'Kings XI Punjab':
        prediction_array = prediction_array + [0,0,1,0,0,0,0,0,0]
    elif batting_team == 'Kolkata Knight Riders':
        prediction_array = prediction_array + [0,0,0,1,0,0,0,0,0]
    elif batting_team == 'Mumbai Indians':
        prediction_array = prediction_array + [0,0,0,0,1,0,0,0,0]
    elif batting_team == 'Rajasthan Royals':
        prediction_array = prediction_array + [0,0,0,0,0,1,0,0,0]
    elif batting_team == 'Royal Challengers Bangalore':
        prediction_array = prediction_array + [0,0,0,0,0,0,1,0,0]
    elif batting_team == 'Sunrisers Hyderabad':
        prediction_array = prediction_array + [0,0,0,0,0,0,0,0,1]
    # Bowling Team
    if bowling_team == 'Chennai Super Kings':
        prediction_array = prediction_array + [1,0,0,0,0,0,0,0,0]
    elif bowling_team == 'Delhi Daredevils':
        prediction_array = prediction_array + [0,1,0,0,0,0,0,0,0]
    elif bowling_team == 'Kings XI Punjab':
        prediction_array = prediction_array + [0,0,1,0,0,0,0,0,0]
    elif bowling_team == 'Kolkata Knight Riders':
        prediction_array = prediction_array + [0,0,0,1,0,0,0,0,0]
    elif bowling_team == 'Mumbai Indians':
        prediction_array = prediction_array + [0,0,0,0,1,0,0,0,0]
    elif bowling_team == 'Rajasthan Royals':
        prediction_array = prediction_array + [0,0,0,0,0,1,0,0,0]
    elif bowling_team == 'Royal Challengers Bangalore':
        prediction_array = prediction_array + [0,0,0,0,0,0,1,0,0]
```

```

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]))

```

Activity 3 :Testing:

Test 1

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

```

}]: 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_5=1)
    print(f'Predicted Score : {score} || Actual Score : 147')

```

Predicted Score : 147 || Actual Score : 147

C:\Users\JAYANTH VARMA\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, RandomForestRegressor was fitted with feature names
warnings.warn(

Test 2

- Batting Team : **Mumbai Indians**
- Bowling Team : **Kings XI Punjab**
- Final Score : **176/7**

```

}]: batting_team='Mumbai Indians'
    bowling_team='Kings XI Punjab'
    score = score_predict(batting_team, bowling_team, overs=12.3, runs=113, wickets=2, runs_last_5=55, wickets_last_5=0)
    print(f'Predicted Score : {score} || Actual Score : 176')

```

Predicted Score : 187 || Actual Score : 176

Test 3

- Batting Team : **Kings XI Punjab**
- Bowling Team : **Rajasthan Royals**
- Final Score : **185/4**

These Test Was done before the match and final score were added later.

```

: batting_team="Kings XI Punjab"
    bowling_team="Rajasthan Royals"
    score =score_predict(batting_team, bowling_team, overs=14.0, runs=118, wickets=1, runs_last_5=45, wickets_last_5=0)
    print(f'Predicted Score : {score} || Actual Score : 185')

```

Predicted Score : 176 || Actual Score : 185

C:\Users\JAYANTH VARMA\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, RandomForestRegressor was fitted with feature names
warnings.warn(

Test 4

- Batting Team : **Kolkata Knight Riders**
- Bowling Team : **Chennai Super Kings**
- Final Score : **172/5**

```

: batting_team="Kolkata Knight Riders"
    bowling_team="Chennai Super Kings"
    score = score_predict(batting_team, bowling_team, overs=18.0, runs=150, wickets=4, runs_last_5=57, wickets_last_5=1)
    print(f'Predicted Score : {score} || Actual Score : 172')

```

Predicted Score : 172 || Actual Score : 172

Milestone 5: Model Deployment

Activity 1: Save the best model

After analyzing the accuracy and mean absolute error of above models we'll select the best model out of those 3 models.

As we can see Random forest is performing well out of all the models.

```
import pickle
filename = "ml_model.pkl"
pickle.dump(forest, open(filename, "wb"))
```

✓ 0.2s

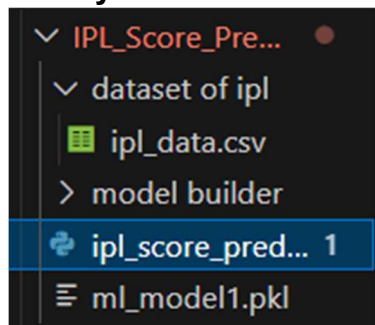
Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built.

We will be using the streamlit package for our website development.

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps.

Activity 2.1: Create a web.py file and import necessary packages:



```
import math
import numpy as np
import pickle
import streamlit as st
```

NOTE : If you get any error like “streamlit not found”. You have to make sure streamlit is installed on your system.

- Running the command “pip install streamlit” would do that job for you.

Activity 2.2: Defining teams names and loading the pipeline:


```

# SELECT THE BATTING TEAM

batting_team= st.selectbox('Select the Batting Team ',('Chennai Super Kings', 'Delhi Daredevils',
'Kings XI Punjab','Kolkata Knight Riders','Mumbai Indians'
,'Rajasthan Royals','Royal Challengers Bangalore',
'Sunrisers Hyderabad'))

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 capitals':
    prediction_array = prediction_array + [0,1,0,0,0,0,0,0]
elif batting_team == 'Punjab kings':
    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]

#SELECT BOWLING TEAM

bowling_team = st.selectbox('Select the Bowling Team ',('Chennai Super Kings', 'Delhi Daredevils', 'Kings XI Punjab',
'Kolkata Knight Riders','Mumbai Indians','Rajasthan Royals',
'Royal Challengers Bangalore','Sunrisers Hyderabad'))

if bowling_team==batting_team:
    st.error('Bowling and Batting teams should be different')
# Bowling Team
if bowling_team == 'Chennai Super Kings':
    prediction_array = prediction_array + [1,0,0,0,0,0,0,0]
elif bowling_team == 'Delhi capitals':
    prediction_array = prediction_array + [0,1,0,0,0,0,0,0]
elif bowling_team == 'Punjab kings':
    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]

```

Activity 2.3: Accepting the input from user and prediction

```

col1, col2 = st.columns(2)

#Enter the Current Ongoing Over
with col1:
    overs = st.number_input('Enter the Current Over',min_value=5.1,max_value=19.5,value=5.1,step=0.1)
    if overs-math.floor(overs)>0.5:
        st.error('Please enter valid over input as one over only contains 6 balls')
with col2:
    #Enter Current Run
    runs = st.number_input('Enter Current runs',min_value=0,max_value=354,step=1,format='%i')

#Wickets Taken till now
wickets =st.slider('Enter Wickets fallen till now',0,9)
wickets=int(wickets)

col3, col4 = st.columns(2)

with col3:
    #Runs in last 5 over
    runs_in_prev_5 = st.number_input('Runs scored in the last 5 overs',min_value=0,max_value=runs,step=1,format='%i')

```

```

with col3:
#Runs in last 5 over
    runs_in_prev_5 = st.number_input('Runs scored in the last 5 overs',
                                     min_value=0,max_value=runs,step=1,format='%i')

with col4:
#Wickets in last 5 over
    wickets_in_prev_5 = st.number_input('Wickets taken in the last 5 overs',
                                       min_value=0,max_value=wickets,step=1,format='%i')

#Get all the data for predicting

prediction_array = prediction_array + [runs, wickets, overs, runs_in_prev_5,wickets_in_prev_5]
prediction_array = np.array([prediction_array])
predict = model.predict(prediction_array)

if st.button('Predict Score'):
    #Call the ML Model
    my_prediction = int(round(predict[0]))

    #Display the predicted Score Range
    x=f'PREDICTED MATCH SCORE : {my_prediction-5} to {my_prediction+5}'
    st.success(x)

```

To run your website go to your terminal with your respective directory and run the command :

- “stream lit run web.py”

```

(base) C:\Users\JAYANTH VARMA>cd aiml
(base) C:\Users\JAYANTH VARMA\aiml>streamlit run app.py

```

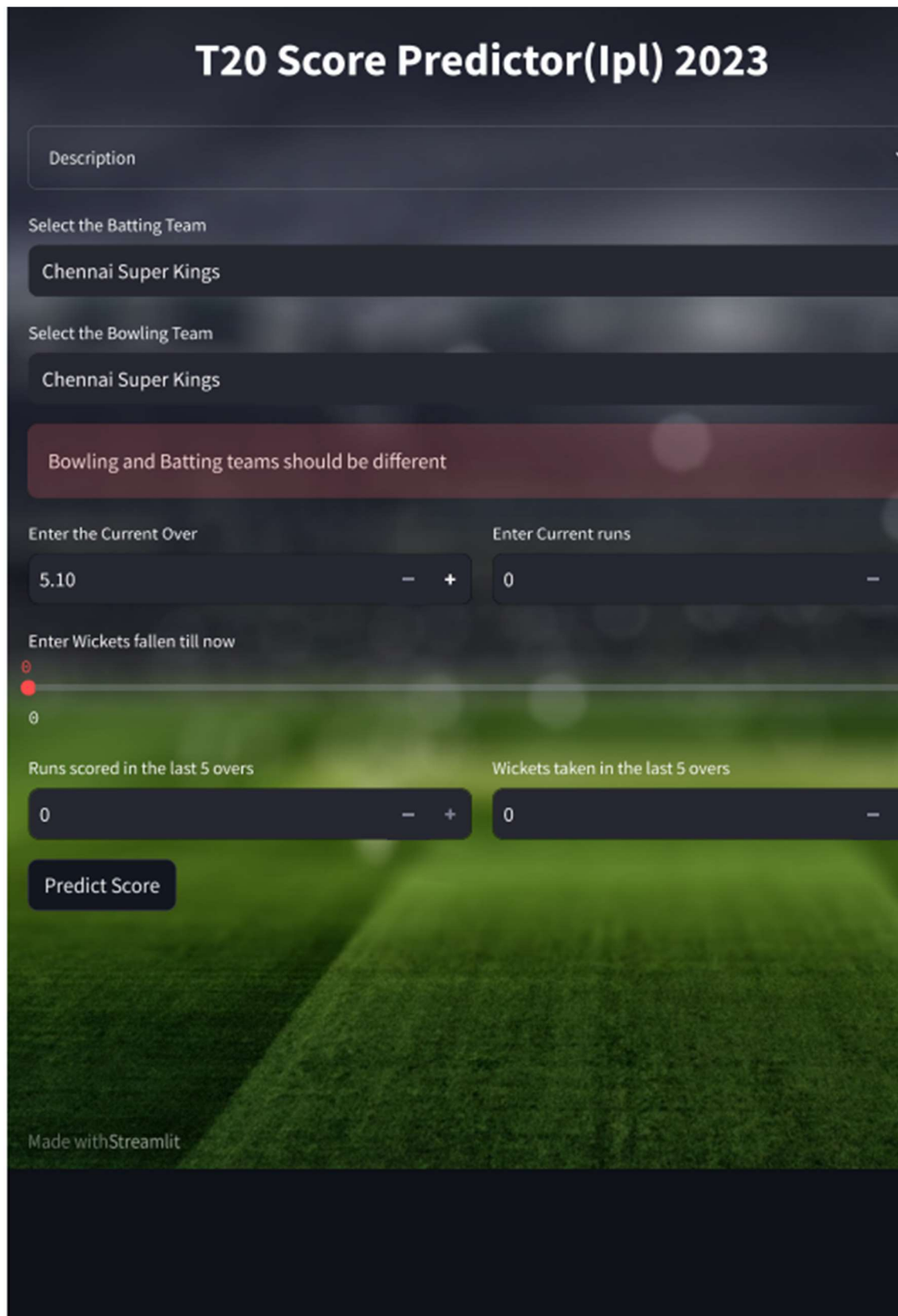
You can now view your Streamlit app in your browser.

Local URL: <http://localhost:8501>
Network URL: <http://192.168.1.5:8501>

On successful execution you'll get to see this in your terminal.

HOW DOES YOUR WEBSITE LOOK LIKE

Before entering the data



The image shows a web application titled "T20 Score Predictor(Ipl) 2023". The interface is dark-themed with a background image of a cricket field. It features several input fields and a "Predict Score" button. A red error message is displayed below the team selection fields.

T20 Score Predictor(Ipl) 2023

Description

Select the Batting Team

Chennai Super Kings

Select the Bowling Team

Chennai Super Kings

Bowling and Batting teams should be different

Enter the Current Over

5.10

Enter Current runs

0

Enter Wickets fallen till now

0

Runs scored in the last 5 overs

0

Wickets taken in the last 5 overs

0

Predict Score

Made with Streamlit

After entering the data :

T20 Score Predictor(Ipl) 2023

Description

A Simple ML Model to predict IPL Scores between teams in an ongoing match. To make sure the model results accurate score and some reliability the minimum no. of current overs considered is greater than 5 overs.

Select the Batting Team

Kolkata Knight Riders

Select the Bowling Team

Chennai Super Kings

Enter the Current Over

8.00

Enter Current runs

70

Enter Wickets fallen till now

2

Runs scored in the last 5 overs

50

Wickets taken in the last 5 overs

1

Predict Score

PREDICTED MATCH SCORE : 165 to 175

Made with Streamlit

Milestone 6: Project Demonstration & Documentation

Activity 1: - Record explanation Video for project end to end solution.

Activity 2: - Project Documentation-Step by step project development procedure.