T20- TOTALITARIAN :MASTERING SCORE PREDICTIONS(IPL)

1.INTRODUCTION:

1.1 PROJECT OVERVIEW:

Cricket score prediction is an area where the first innings score of a cricket match is predicted using some techniques. There are various systems and prediction methods used to predict the cricket score of the ODI and the T20 cricket matches. CRR method is widely used to predict the score of the cricket match. In the CRR method, the number of runs scored in an over is multiplied by the total number of overs in an innings. This method only focuses on the runs made in an over but it does not focus on the different parameters. This method can only predict the score based on the current score and not based on the various parameters. We are working to improve on the predictions by considering different parameters and to improve the accuracies of the existing systems**.**

KEY COMPONENTS:

- Data collection
- Preprocessing
- Feature engineering
- ML algorithms
- Real time processing
- Predict output

Cricket, with its ever-evolving dynamics, offers an exhilarating experience for fans worldwide, especially in the context of T20 matches known for their fast-paced and unpredictable nature. This introduction outlines the motivation behind the T20 Score Prediction ML Model and App project, highlighting its relevance in the realm of sports analytics and entertainment. Our model overview provides a foundational understanding of the methodologies

employed, key components, and its role within the comprehensive T20 Score Prediction system. Subsequent sections will delve into the technical details, real-time processing capabilities, and user interaction aspects of the model. Our aim is to predict the first innings score of the live IPL match. We have studied the systems which predict the scores of the cricket match. Most of the systems focus on the current run rate. We will be predicting the score of an IPL match by considering various factors like runs scored, overs bowled, wickets taken, etc. The reason behind selecting these features is that we need to build a model that can understand the dynamicity of the cricket game. Therefore we are considering the factors which will be focusing on the dynamic nature of the cricket game.

1.2 PURPOSE:

The purpose of the T20 Score Prediction ML Model in the broader context of the T20 Score Prediction system is multi-faceted, aiming to bring value to various stakeholders within the cricket ecosystem. The primary purposes include:

- Enhanced viewer Experience
- Support strategic planning of teams
- Building a community of cricket Enthusiasts
- Provide real time predictions
- Enable Data-Driven and Decision Making etc
- Fair odds for bookmakers
- Educate users through insights
- Contribute to sports analytical Advancements
- Create a fair and Transparent Predictive Environment

The overarching purpose is to create a comprehensive platform that not only predicts T20 match scores but also contributes to the development of a knowledgeable and engaged community, supports cricket teams, and fosters fair play within the betting industry.

2.LITERATURE SURVEY:

2.1 EXISTING PROBLEM:

The existing problems in the context of T20 cricket and sports prediction models can include challenges related to accuracy, transparency, and ethical considerations. Here are some common existing problems:

- Prediction Accuracy
- Data quality and Timeliness
- Ethical consideration in sports Betting
- Regulatory compliance challenges

Addressing these existing problems is crucial for the development and deployment of an effective T20 Score Prediction ML Model and App. The proposed solution should aim to mitigate these challenges and offer a more accurate, transparent, and engaging experience for users and stakeholders.

2.2 REFERENCES:

1. R. P. Ram Kumar, P. Sanjeeva, S. F. Lazarus, D. V. Krishna, Intl. J. Inno. Tech.

Explor. Engg 8, 11S2 (2019)

2. M. N. Mohammad, Ch. U. Kumari, A. S. D. Murthy, B. O. L. Jagan, K. Saikumar,

Mater. Today Proc 45, 3 (2021)

https://www.ijcrt.org/

https://www.e3s-conferences.org/articles/e3sconf/abs/2023/67/e3sconf_icmpc2023_01053/e3sconf_icmpc2023_01053.html

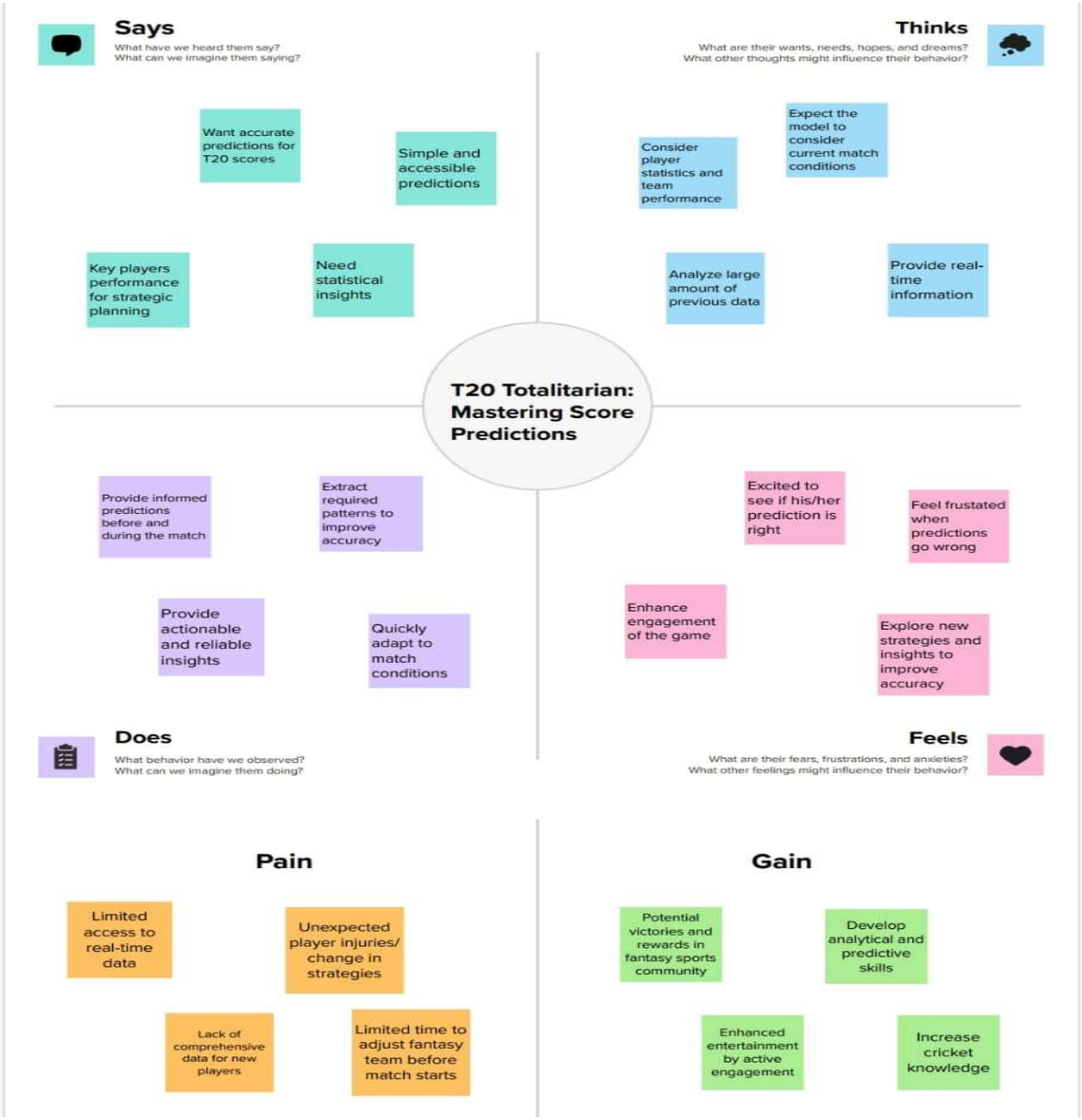
2.3 PROBLEM STATEMENT

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 stratergy or wrong calculation.The models are not user-friendly interface not real time models

3.IDEATION & PROPOSED SOLUTION

3.1EMPATHY MAP CANVAS-

An empathy map is a powerful visual tool that captures a product team'sknowledge of a certain type of user's thoughts, feelings, and actions.This tool helps product teams build a broader understanding of the "why"aspect behind user needs and wants. As a team identifies what they know about the user and places this information on a chart, they gain a moreholistic view of the user's world and his or her problems, or opportunityspace.



3.2IDEATION AND BRAINSTORMING-

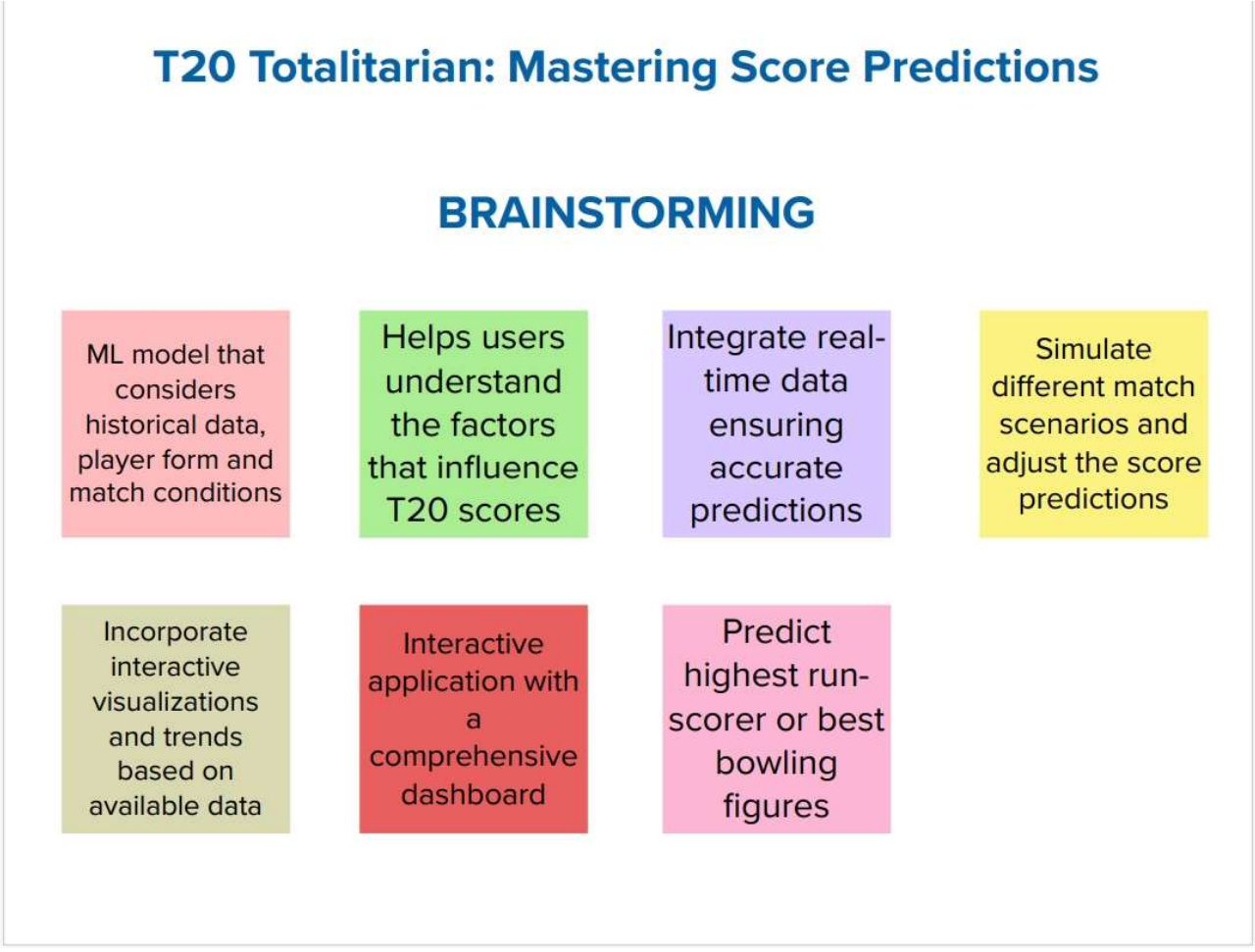
Purpose of Brainstorming

• To find innovative solutions to problems

• To leverage creativity and motivate to higher plateau of thinking

• Create the opportunity for expression of uncultivated ideas

• To draw from the diversity of job skills, responsibilities, personalities, education's and backgrounds



4.REQUIREMENT ANALYSIS:

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.
- Link : https://code.visualstudio.com/download
- Create Project:
    1. Open VS Code.
    2. Create a Folder and name it "T20_Score_Predictor" .
- Machine Learning Concepts o

  Linear Regression: https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86

  XGBRegressor : https://medium.com/@aryanbajaj13/prediction-using- xgb-regressor-using-python-3-ad1a57e105af

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/

4.1FUNCTIONAL REQUIREMENTS:

1. User Registration and Authentication: Users should be able to register and create accounts securely. Authentication mechanisms to ensure secure access to the app.

2. Real-Time Score Predictions: The ML model must provide real-time predictions for T20 cricket match scores. Predictions should be updated dynamically during live matches.

3. User Interface (UI): Intuitive and responsive UI for the mobile app. Interactive features for users to view predictions, historical data, and participate in discussions.

4. Community Engagement: Forums and discussion boards for users to engage in cricket-related discussions. Prediction leagues and challenges to encourage user participation.

5.Educational Content: Integration of educational materials explaining sports analytics, prediction models, and cricket statistics. User-friendly access to learning resources within the app.

4.2NON FUNCTIONAL REQUIREMENTS

1. Performance: The app should provide real-time predictions without significant delays. Response times for user interactions should be minimal.

2. Reliability: The ML model should consistently deliver accurate predictions.

The app should be reliable and available during live matches.

3. Usability: The UI should be user-friendly, catering to users with varying levels of cricket knowledge. Navigation should be intuitive, and features should be easily accessible.

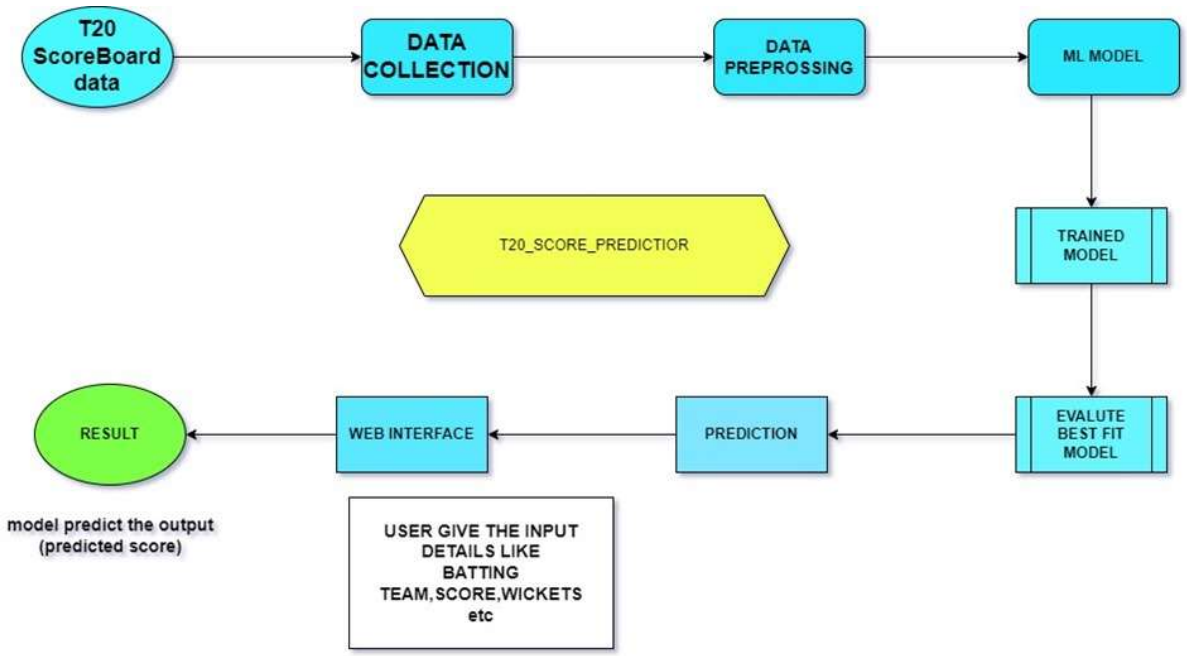
4. Data Privacy: Implement data privacy measures in accordance with relevant regulations. Clearly communicate the app's privacy policy to users.

5. Compatibility: The app should be compatible with a variety of devices and operating systems. Ensure seamless performance on both Android and iOS platforms.

5.PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS&USER STORIES

A Data Flow Diagram (DFD) visually represents the flow of data within a system. In the context of a T20 score prediction ML model, the steps involved can be illustrated through a DFD. Here's a simplified representation. This DFD provides a high-level overview of the key steps involved in T20 score prediction, from data processing to real-time predictions, and user interaction. Depending on the complexity of your system, you may expand each step into more detailed processes and data flows.



User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Directly interact with model	USN-1	As a customer, I w to get unt an estimate the prediction of e scor the end of innings at	1. I should be able to input the batting team Bowling team, score, venue, wickets left and overs etc 2.System will provide predicted score at end of innings	High	Sprint-1
		USN-2	As a user, I want to receive timely notifications or updates during live matches so that I stay informed abo changing an predictio and match it ns dynami s	1. I should understand how should predicted score will be changed.	medium	Sprint-1
Customer (Web user)	Directly interact with model	USN-3	As a user, I want an intuitive hat and userfriendly interface so I and can easily navigate interact with the prediction model.	1. I should be able to input the batting team Bowling team, score, venue, wickets left and overs etc 2.System will provide predicted score at end of innings	high	Sprint-2

Admin	Upload historical T20_score data and graphs	USN-4	As an admin, I wa to ıt l upload historica T20_score data to improve predictive accuracy	1. I should be able to upload CSV files containing past T20_score data. 2. The system should process and store this data for model training. 3. A confirmation of successful upload should be provided	HIGH	SPRINT-1
Data an lyst	View prediction accuracy	USN-5	As a data analyst, I want to view the accuracy of shipping predictions.	1. The system should provide a dashboard with prediction accuracy metrics. 2. Accuracy metrics should include measures like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).	MEDIUM	SPRINT-1
Cricket Ieam Analyst:	Understanding The strategies of match	USN-6	As a cricket team analyst, I want detailed insigl opposing team strategies and ke into player performan y ces to assist our tea ı in strategic planning and decision-making.	1.The data analyst should be able to access this information easily. 2. I while change only bowling team and predict the strength of batting team from the application	MEDIUM	SPRINT_2

5.2 SOLUTION ARCHITECTURE

Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	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 stratergy or wrong calculation.The models are not user-friendly interface not real time models.
2.	Idea / Solution description	we propose the development of an advanced T20 score prediction ML model, accompanied by a user-friendly interface, seamless integration capabilities, and a robust ethical framework. 1.By developing a good model with different types of machine learning algorithms, incorporating ensemble methods to ensure accurate and robust predictions. 2.By developing APIS for seamless integration with existing cricket analytical tools. 3. To design the model to be scalable, ensuring optimal performance with increased data volume and user load.

3.	Novelty / Uniqueness	The uniqueness of the proposed T20 score prediction ML model lies in its combination of advanced features, seamless integration, user-centric design, and ethical considerations and continuous improvement and updatation of data.scalable for everyone,safe and secure and accurate data present in the model. The inclusion of educational content within the interface enhances user understanding of cricket analytics and prediction models. Gamification elements add an element of excitement, keeping users engaged and invested in the prediction experience.T he model prioritizes fairness, transparency, and ethical considerations
4.	Social Impact / Customer Satisfaction	Our solution will have a significant positive impact on the cricket lovers and the people who watches cricket. 1..Enhanced Viewer Experience-Accurate predictions provide cricket enthusiasts with a more engaging and immersive viewing experience, creating excitement and anticipation during T20 matches. 2.Educational opportunities-The inclusion of educational content within the interface offers opportunities for users to learn more about cricket analytics, statistical factors influencing scores, and the functioning of prediction models, fostering a greater understanding of the sport. 3.Empowerment Through Information-Coaches and players can benefit from insights into opposing team strategies and key player performance, empowering them to make more informed decisions and strategies during matches 4.Inspiration for Innovation-he development of sophisticated prediction models in sports can inspire innovation in other fields.
5.	Business Model (Revenue Model)	The revenue model for our project involves partnering with e-commerce businesses,sports businesses, different types of channels to integrate this model to the people very fast. Revenue through the advertisements on the platform targeting a larger userbase.By partnering with data providers,cricket teams and leagues, funding companies for investment and bookmakers. By Offering a preemium model with basic predictions and premium

		subscription tiers for advanced features. By Providing responsive customer support to address user queries, issues, and feedback
6.	Scalability of the Solution	Our solution is highly scalable and can accommodate the growth of cricket sport. The machine learning model can handle large volumes of data, and its architecture is designed to be easily expandable to meet increasing demands. This scalability is crucial for accommodating the rapid growth of the e-commerce industry. The T20 score prediction ML model can handle growth in terms of users, data, and features while maintaining optimal performance and user satisfaction. Regular monitoring, testing, and optimization are essential components of a scalable solution.

6.PROJECT PLANNING &SCHEDULING

6.1TECHNICAL ARCHITECTURE:

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Programming Languages used	Programming Language to understand our commands	Java / Python
3.	Execution of code	To Execute the code an to create a model	Jupyter Notebook / Google colab for execution of the code
4.	Web Interface	To create the form and connect it to the backend data	Streamlit for the webinterface,forms,webdevelopment /flask module
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	DataSet collection	Dataset,no of rows and columns,data type	Kaggle / Github/Google
7.	File storage	File storage requirements	Stored in Internal memory of system
8.	Python Modules-1	For preprocessing the data from data set, for visuliazation and analysis of data	Numpy,Pandas,Seaborn,
9.	Python Modules-2	Model building and implementation of machinelearning algorithms like classification,regression,decision tree,Random forest etc.	Sklearn library/XGB Boost

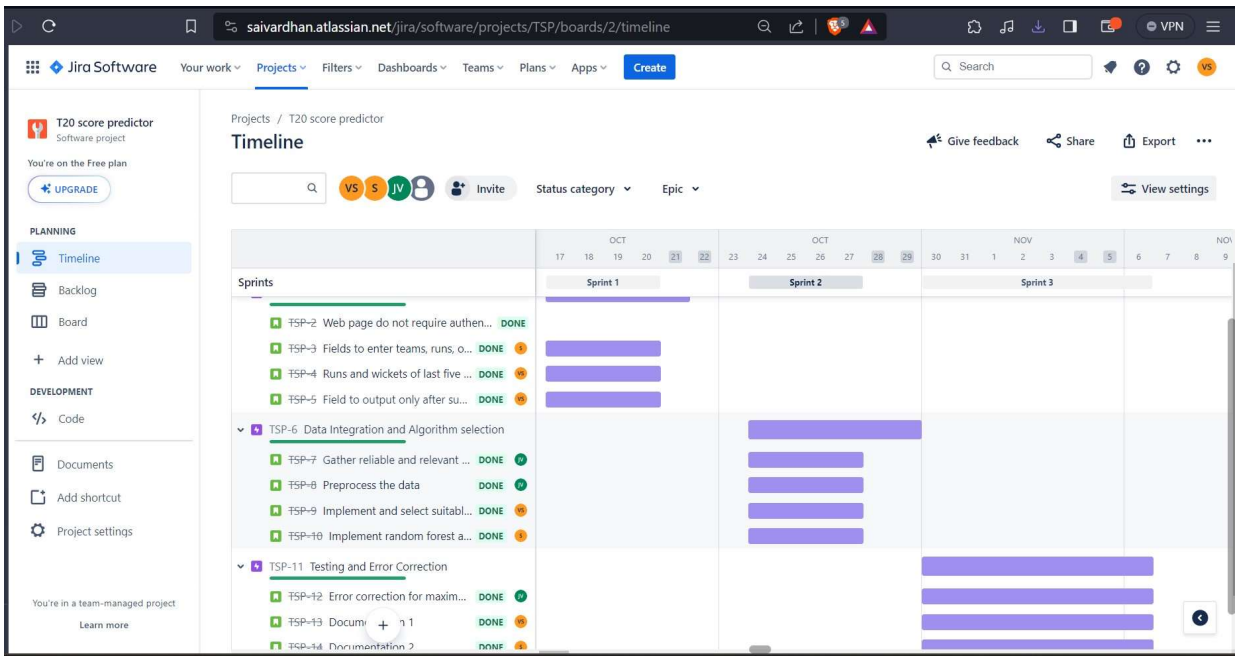
10.	Machine Learning Algorithms	Random forest algorithm used due to its high accuracy,robustness,feature importance , versatility to the model as it performs bagging technique.XGB Reggresor is used for regression problems where the intent is to predict continuous numericalvalues.	Random Forest and XGB Regressor

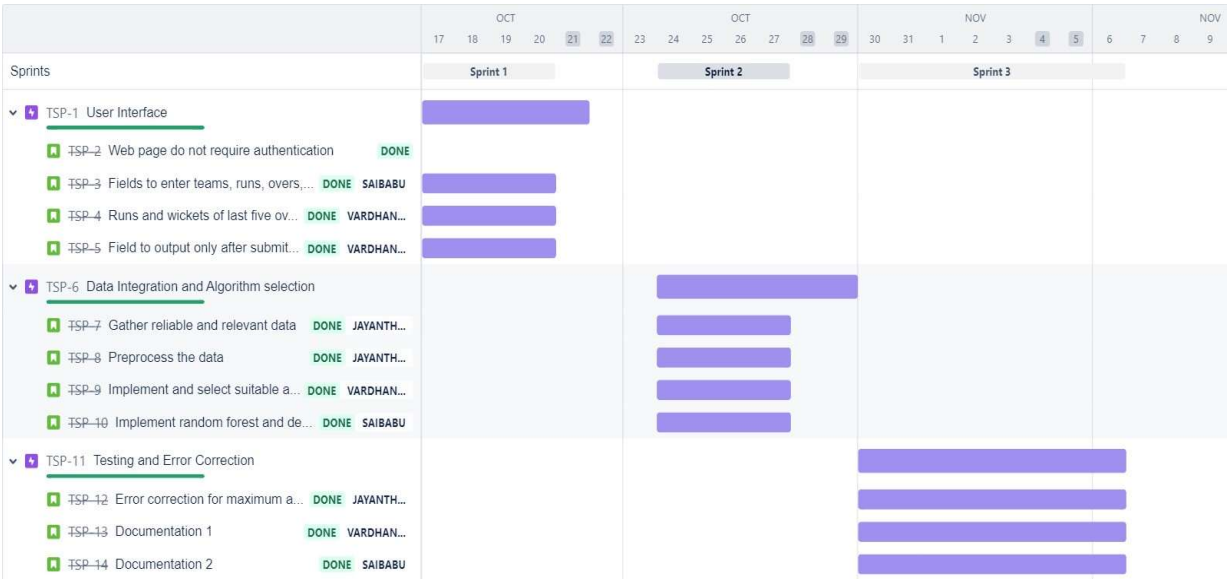
Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Sklearn is most usefuland robust framework for machine learning in python.Tensorflow is a library for machine learning tasks.Xg Boost is a open source framework for algorthms like regression,classification.	Sklearn, Tenosrflow,Keras,Xg Boost
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Streamlit
S.No	**Characteristics**	**Description**	**Technology**
4.	Availability	Available for everyone as a application	Html,css for userinterface,Streamlit or flask
5.	Performance	The accuracy of the model is 93 percent.	Random forest Algorithm,xb boost

6.2 SPRINT PLANNING &ESTIMATION

JIRA SOFTWARE SPRINT TIMELINE SCREENSHOTS





Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Interface		Fields to enter teams, runs , overs and wickets	2	Low	Sai Babu
Sprint-1			Fields for runs and wickets of last five overs	2	Low	Vardhan
Sprint-2	Data Integration & Algorithm Selection		Reliable data Collection	1	High	Jayanth
Sprint-2			Preprocessing the data	2	Medium	Jayanth
Sprint-1	User Interface		Output field on web page available only after submitting	1	High	Vardhan
Sprint-2	Data Integration & Algorithm Selection		Implementation and selection of suitable algorithm	2	High	Vardhan
Sprint-2			Implementation of Random forest and decision tree algorithm	1	High	Sai babu
Sprint-3	Testing and Error Correction		Achieving maximum accuracy while having no or minimum errors	1	High	Jayanth
Sprint-3			Documentation phase 1	2	Medium	Vardhan
Sprint-3			Documentation Phase 2	2	Medium	Sai babu

6.3SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	5	5 Days	17 Oct 2023	21 Oct 2023	5	21 Oct 2023
Sprint-2	6	6 Days	24 Oct 2023	29 Oct 2023	6	30 Oct 2023
Sprint-3	5	8 Days	31 Oct 2023	07 Nov 2023	5	9 Nov 2023

Burndown Chart

7 CODING AND SOLUTIONING

7.1 Feature 1

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 AND Load the dataset

Import Necessary Libraries

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

Load the dataset

```
[ ]  #Importing dataset
     ipl_df = pd.read_csv('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

Activity-3: Preprocessing the data

```
# Information about Each Column
ipl_df.info()
```

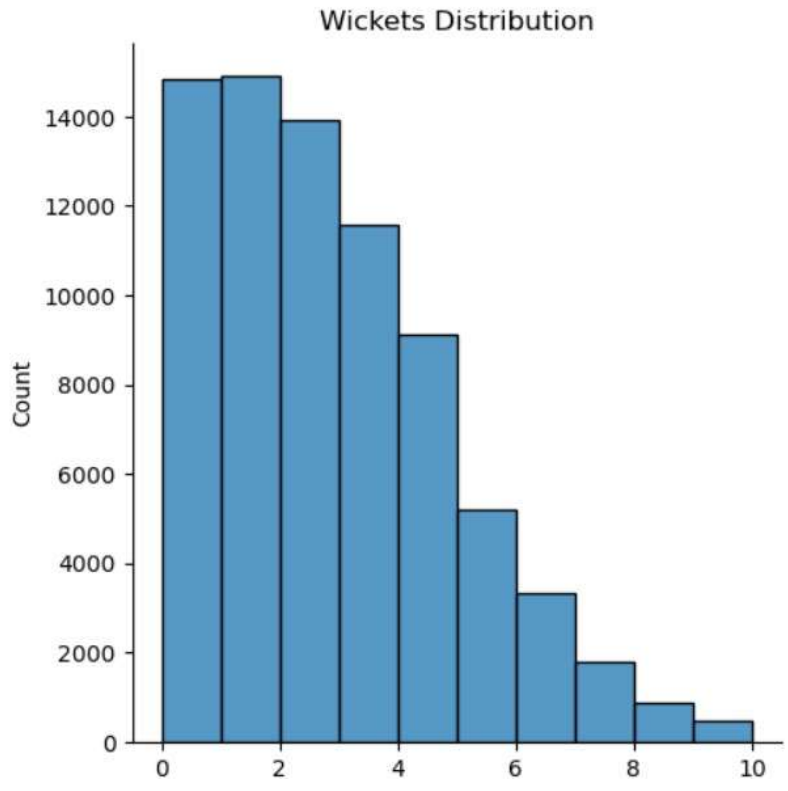
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76014 entries, 0 to 76013
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   mid             76014 non-null  int64
 1   date            76014 non-null  object
 2   venue           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
```

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

```
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
```

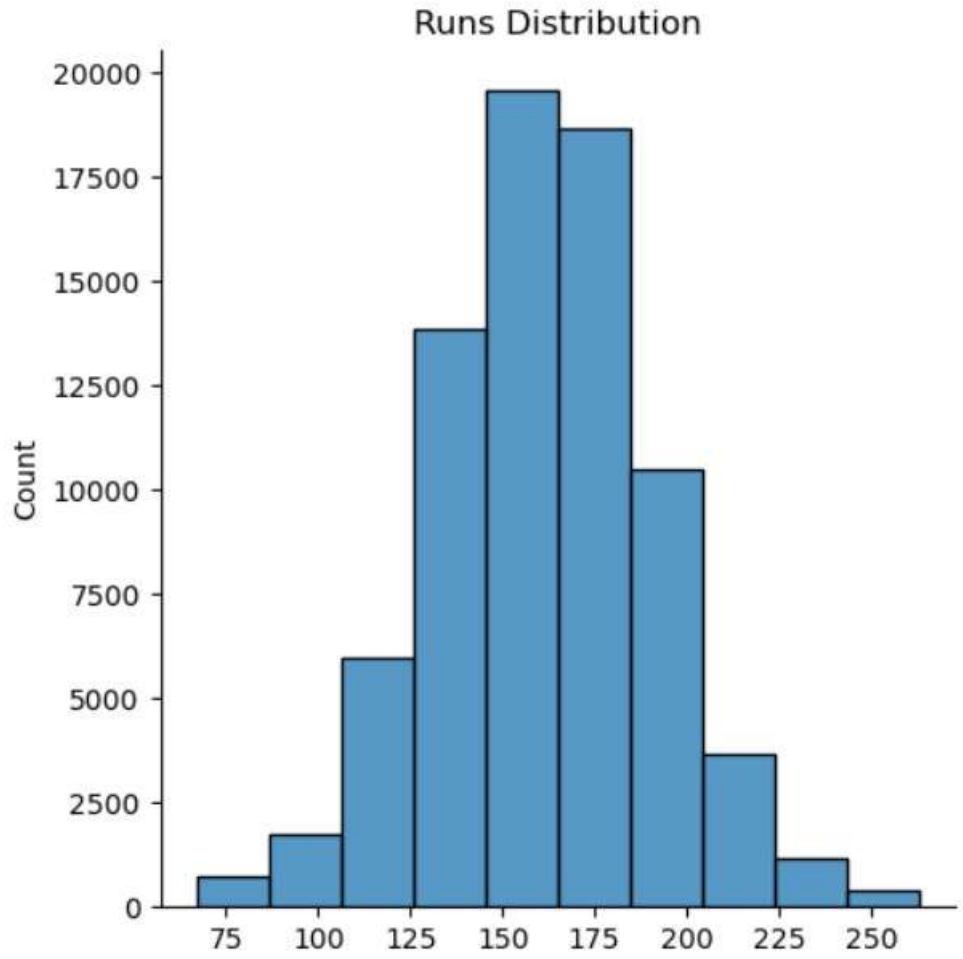
```
#Wickets Distribution
sns.displot(ipl_df['wickets'],kde=False,bins=10)
plt.title("Wickets Distribution")

plt.show()
```

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



```
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:1
  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']
```

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 4: Splitting data into train and test sets

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)
```

Activity 5: Model Evolution and Model training

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(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))))
```

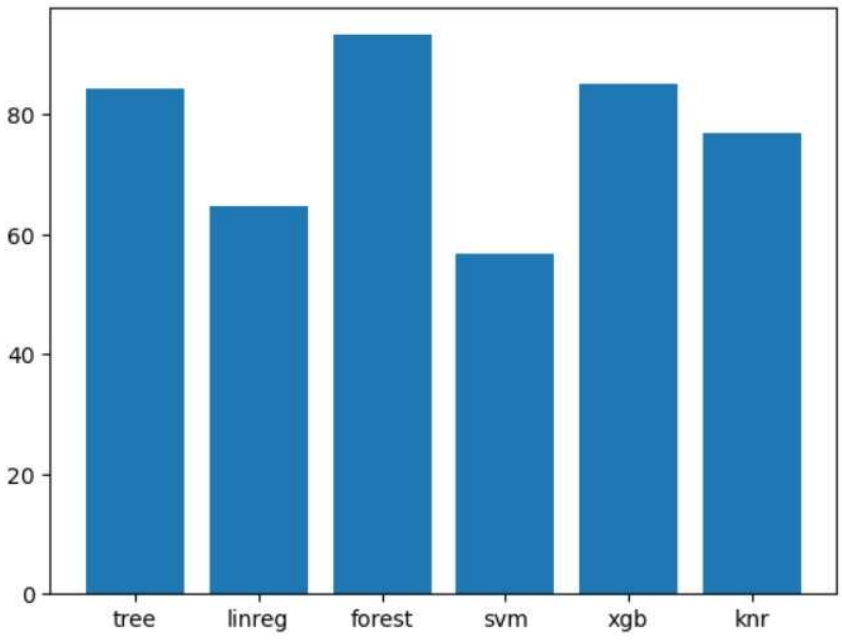
```
---- KNR - Model Evaluation ----
Mean Absolute Error (MAE): 9.900349040139616
Mean Squared Error (MSE): 203.2539267015707
Root Mean Squared Error (RMSE): 14.256715144154725
```

Activity 6: 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>
```



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

```
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]))
```

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 n
andomForestRegressor 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
andomForestRegressor 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
```

7.2 Feature 2

Activity-1:Model Deployment(Stream lit application)

Export Model ¶

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

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

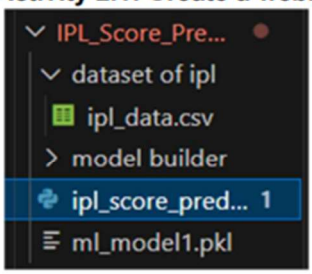
Activity 2: Integrate with Web Framework

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.

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

#SET PAGE WIDE
st.set_page_config(page_title='IPL_Score_Predictor',layout="centered")

#Get the ML model

filename="C:\\Users\Sai\Downloads\ml_model.pkl"
model = pickle.load(open(filename,'rb'))

#Title of the page with CSS

st.markdown("<h1 style='text-align: center; color: white;'> IPL Score Predictor 2022 </h1>", unsafe_allow_html=True)

#Add background image

st.markdown(
        f"""
        <style>
        .stApp {{
            background-image: url("https://4.bp.blogspot.com/-F6aZ
            F5PMwBQ/Wrj5h204qxI/AAAAAAAABao/4QLn48RP3x0P8Ry0CcktxilJqRfv1IfcACLcBGAs/
            s1600/GURU%2BEDITZ%2Bbackground.jpg");
            background-attachment: fixed;
            background-size: cover
        }}
        </style>
        """,
        unsafe_allow_html=True
    )

#Add description

with st.expander("Description"):
    st.info("""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


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'))
```

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

Activity 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]
```

```
    bowling_team=bowling_team.drop('batting_team')
# BOWLING TEAM
    st.write('Bowling and Batting teams should be different teams')
if bowling_team==batting_team:
                                                       ,'Royal Challengers Bangalore','Sunrisers Hyderabad'))
prediction_array = prediction_array + [0,0,0,0,0,0,0,1]
    prediction_array = prediction_array + [0,0,0,0,0,0,1,0]
elif bowling_team == 'Royal Challengers Bangalore':
    prediction_array = prediction_array + [0,0,0,0,0,1,0,0]
elif bowling_team == 'Rajasthan Royals':
    prediction_array = prediction_array + [0,0,0,0,1,0,0,0]
elif bowling_team == 'Mumbai Indians':
    prediction_array = prediction_array + [0,0,0,1,0,0,0,0]
elif bowling_team == 'Kolkata Knight Riders':
    prediction_array = prediction_array + [0,0,1,0,0,0,0,0]
elif bowling_team == 'Punjab Kings':
    prediction_array = prediction_array + [0,1,0,0,0,0,0,0]
elif bowling_team == 'Delhi Capitals':
    prediction_array = prediction_array + [1,0,0,0,0,0,0,0]
elif bowling_team == 'Chennai Super Kings':
# 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'))

#EFFECT BOWLING TEAM
```

Activity 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”

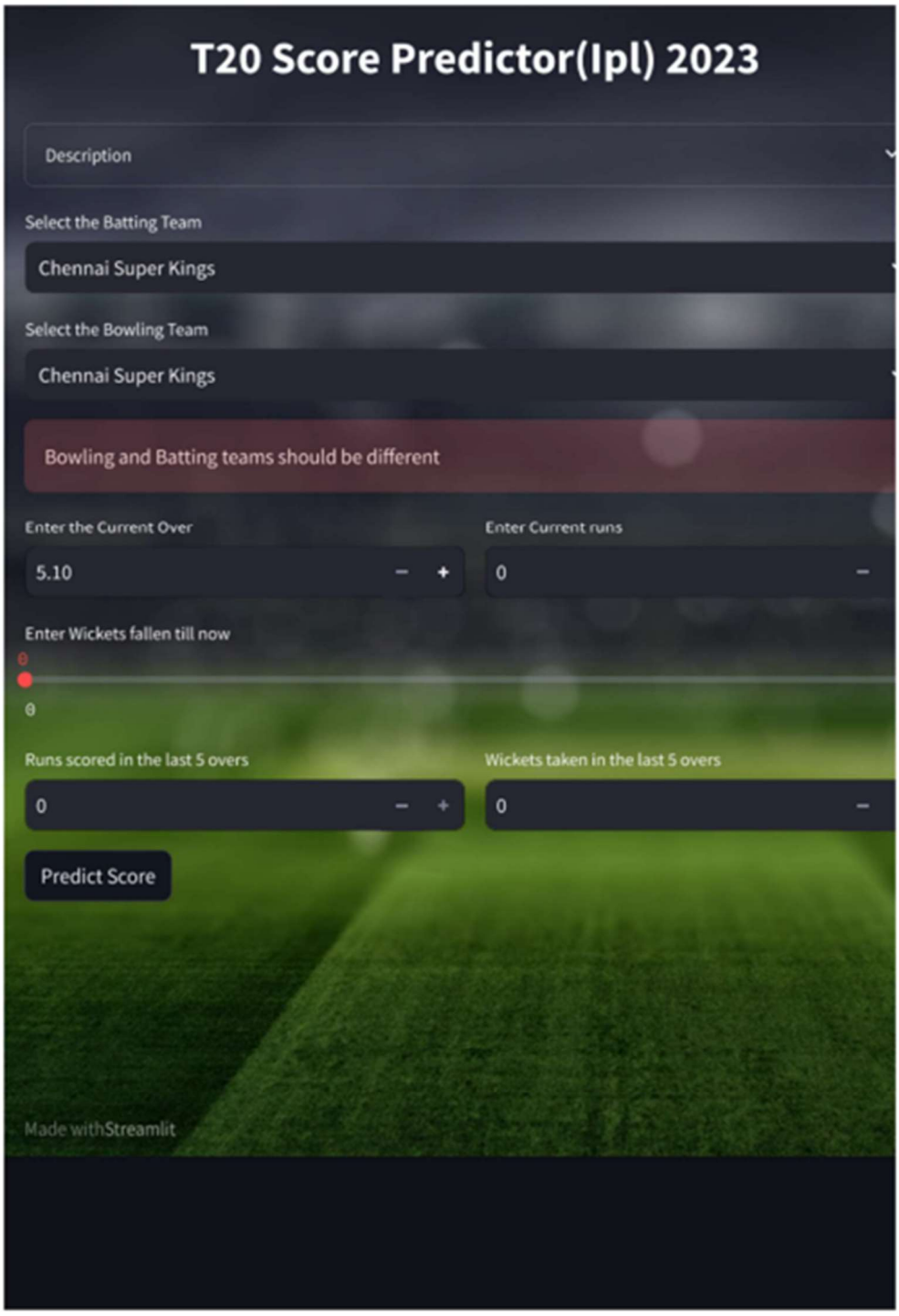
```
(base) C:\Users\JAYANTH VARMA>cd aiml

(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

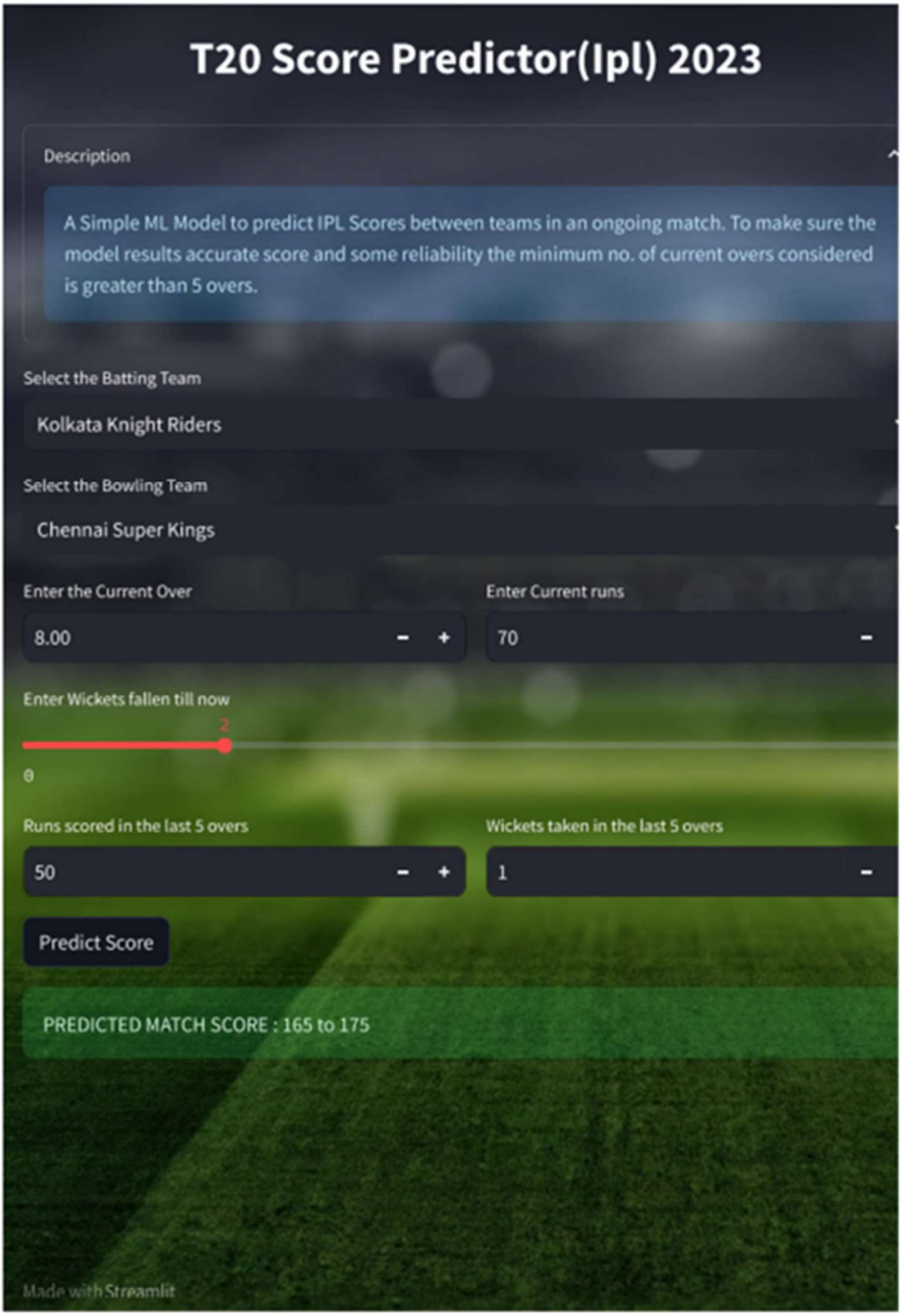
OUTPUT:

HOW DOES YOUR WEBSITE LOOK LIKE

Before entering the data



After entering the data :



Predict the score after first innings :165 to 175

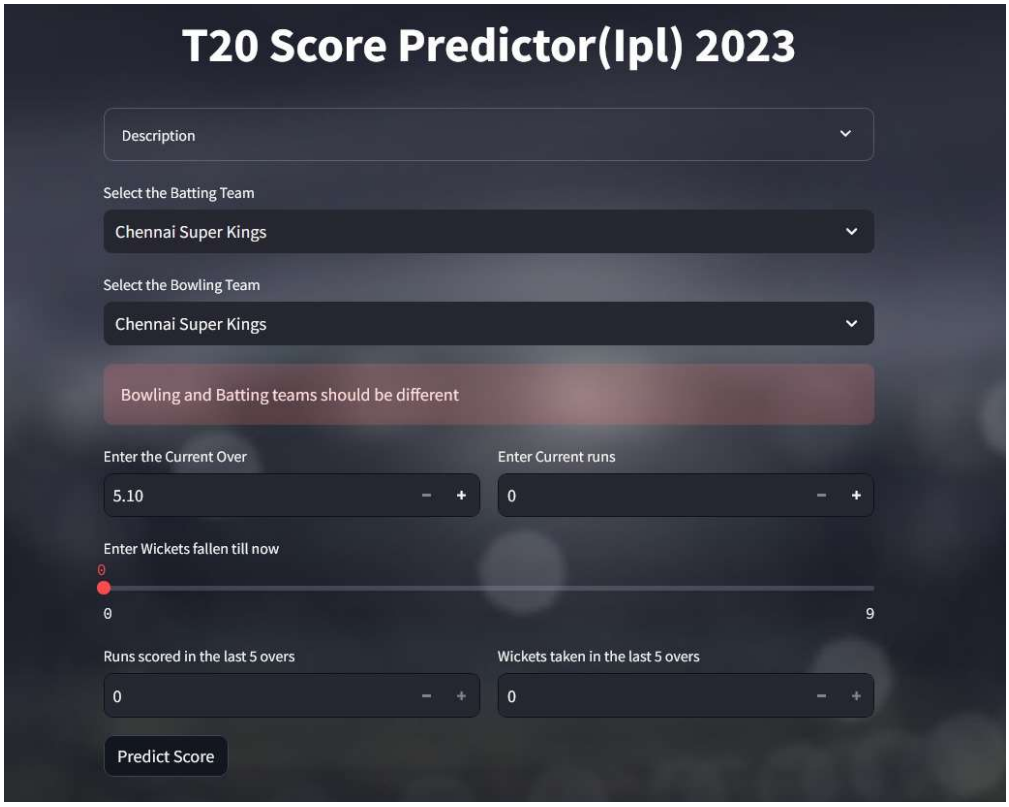
8.MODEL PEFRFORMANCE TESTING

Model Performance Testing:

S No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** **1.Decesion tree regressor** MAE – 4.137309897781102 MSE --137.6454437796061 RMSE -- 11.732239504016533 **2.Linear Regression-** MAE-13.231253750636858 MSE-311.2042685497275 RMSE-17.64098264127391 **3.Random Forest-** MAE- 4.5038041368379815 MSE-59.72223492599992 RMSE-7.7280162348431904 **4.KNeighborsRegressor** MAE-7.7280162348431904 MSE-203.2539267015707 RMSE14.256715144154725	1. ---- Decision Tree Regressor - Model Evaluation ---- Mean Absolute Error (MAE): 4.137309897781102 Mean Squared Error (MSE): 137.6454437796061 Root Mean Squared Error (RMSE): 11.732239504016533 2. ---- Linear Regression - Model Evaluation ---- Mean Absolute Error (MAE): 13.231253750636858 Mean Squared Error (MSE): 311.2042685497275 Root Mean Squared Error (RMSE): 17.64098264127391 3. ---- Random Forest Regression - Model Evaluation ---- Mean Absolute Error (MAE): 4.5038041368379815 Mean Squared Error (MSE): 59.72223492599992 Root Mean Squared Error (RMSE): 7.7280162348431904 4. ---- KNR - Model Evaluation ---- Mean Absolute Error (MAE): 9.900349040139616 Mean Squared Error (MSE): 203.2539267015707 Root Mean Squared Error (RMSE): 14.256715144154725
2.	Tune the Model	Hyperparameter Tuning – MAE-14.763365662866732 MSE-383.20455178342854 RMSE-19.57561114712459 Validation Method – SVMSUPPORT VECTOR MACHINE	---- Support Vector Regression - Model Evaluation ---- Mean Absolute Error (MAE): 14.763365662866732 Mean Squared Error (MSE): 383.20455178342854 Root Mean Squared Error (RMSE): 19.57561114712459 1.

9.RESULTS:

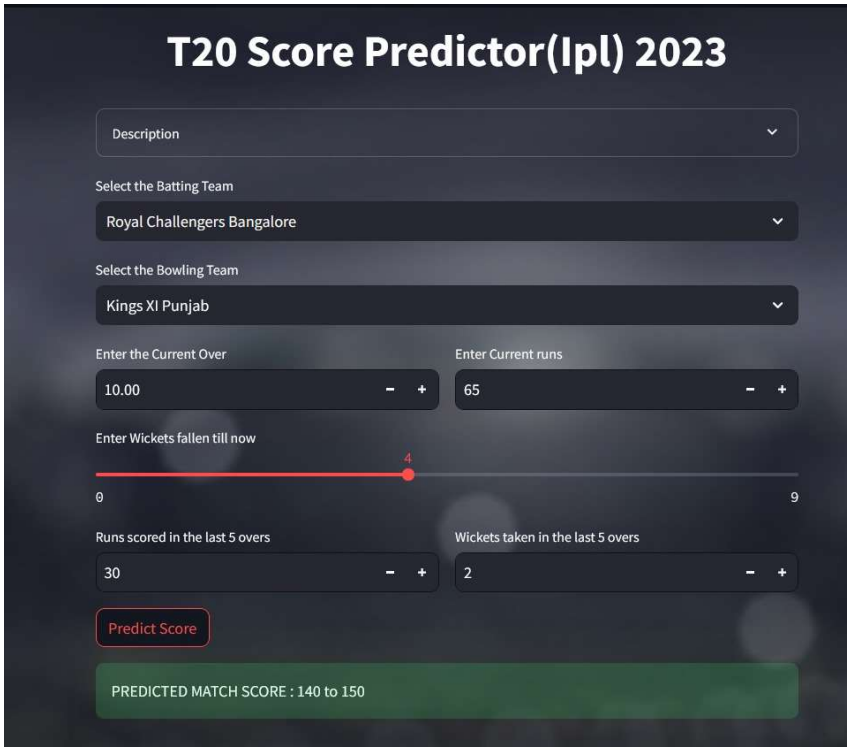
Before entering data:



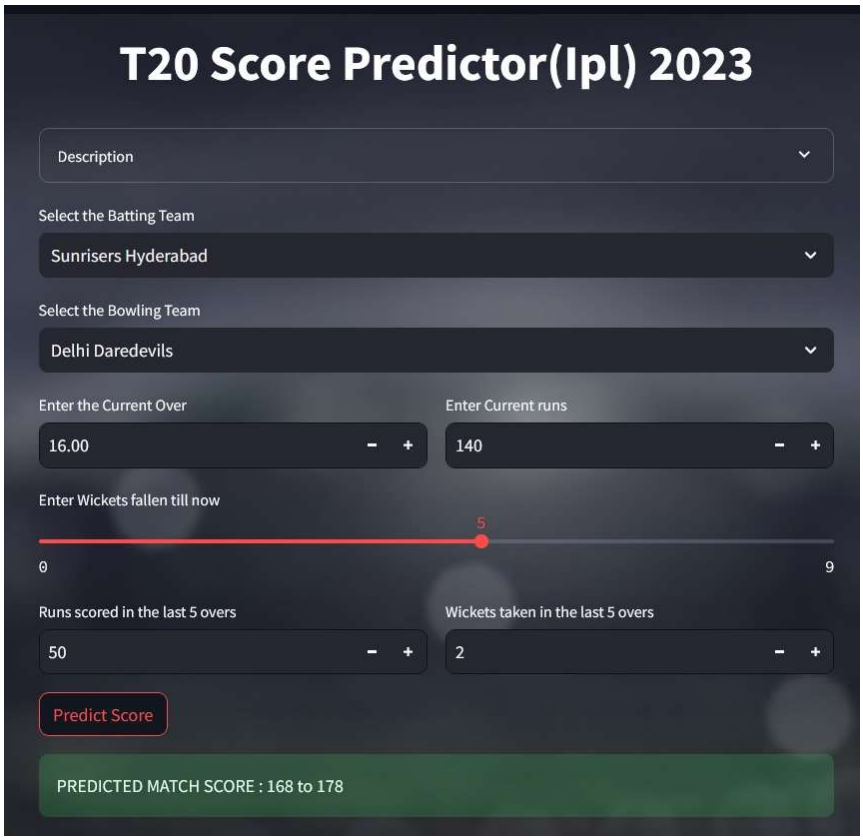
After entering data:

1st case:

2nd case:



3rd case:



10.ADVANTAGES AND DISADVANTAGES

A The T20 score prediction ML model has both advantages and disadvantages. Here's an overview of some potential benefits and drawbacks:

Advantages:

1. Enhanced Viewer Experience:

Accurate predictions can enhance the viewing experience for cricket enthusiasts, adding an element of excitement and engagement during T20 matches.

2. Strategic Planning for Teams:

Cricket teams can benefit from insights into opposing team strategies and key player performances, aiding in strategic planning and decision-making.

3. Improved Odds for Bookmakers:

Bookmakers can set more accurate odds, leading to a fairer and more transparent betting environment. This can attract a larger user base and increase trust.

4. Educational Opportunities:

The model can serve as an educational tool, providing users with insights into cricket analytics, statistical factors influencing scores, and the functioning of **prediction models.**

5. Community Engagement:

Features like forums and discussions can foster a sense of community among cricket enthusiasts, allowing them to share predictions and insights.

6. Data-Driven Decision-Making:

- Coaches and players can make more informed decisions based on insights into opposing team strategies and key player performances.

Disadvantages:

1. Prediction Accuracy Challenges:

- Predicting the outcome of sports events, especially dynamic ones like T20 matches, is inherently challenging. The model may not always achieve high accuracy due to the unpredictable nature of the sport.

2.Dependency on Data Quality:

- The accuracy of predictions is highly dependent on the quality and timeliness of the data used for training the model. Inaccurate or outdated data can lead to less reliable predictions.

3. Ethical Considerations:

- Predictive models in sports betting can raise ethical concerns, particularly related to responsible gambling. It's important to implement safeguards to prevent misuse.

4. Overemphasis on Data:

- There's a risk of overemphasizing statistical data and neglecting other important factors, such as player form, team morale, and external conditions, which can also influence match outcomes.

5. User Dependence on Predictions:

- Users might become overly dependent on predictions, potentially affecting the enjoyment of the game and the acceptance of unpredictability in sports.

6. Handling Unexpected Events:

- The model may struggle to handle unexpected events during a match, such as player injuries or sudden changes in team composition, which can significantly impact predictions.

It's crucial to address these challenges carefully, continuously refine the model, and consider user feedback and ethical considerations in the development and deployment of the T20 score prediction ML model.

11.CONCLUSION:

In conclusion, the development of a T20 score prediction ML model and its associated app represents a promising venture with the potential to significantly impact the world of cricket analytics and sports engagement. The project combines the power of machine learning, real-time data processing, and community interaction to create an innovative platform that caters to various user groups, including cricket enthusiasts, teams, and bookmakers.

The advantages of the model include its ability to enhance the viewing experience for cricket fans, provide strategic insights for teams, and contribute to fairer odds in the betting industry. The inclusion of educational content and community engagement features further enriches the user experience and fosters a sense of community among sports enthusiasts.

However, it's crucial to acknowledge the challenges and limitations inherent in predicting sports outcomes, particularly in a dynamic and unpredictable game like T20 cricket. The accuracy of predictions depends on the quality of data, and ethical considerations related to responsible gambling and user dependency on predictions must be addressed.

Looking ahead, the future scope of the project is expansive. Opportunities for improvement and expansion include refining prediction models, exploring new features and factors, embracing real-time analytics, and expanding the model's applicability to other sports. Additionally, advancements such as blockchain integration for transparency, AI-driven insights for coaches, and partnerships with sports organizations can elevate the project's impact.

Continued collaboration with users, stakeholders, and industry experts, along with a commitment to transparency, fairness, and ethical use of AI, will be essential for the sustained success of the T20 score prediction ML model project. The project holds the potential to not only enhance the sports-watching experience but also contribute to the broader field of sports analytics and predictive modelling.

12. FUTURE SCOPE

The future scope of a T20 score prediction ML model project could involve various advancements and expansions, considering emerging technologies, user demands, and industry trends. Here are some potential future directions for the project

1)**Improved Prediction Models:** Continuously enhance the machine learning algorithms and models to improve prediction accuracy. Consider incorporating advanced techniques such as deep learning or reinforcement learning.

2)Feature Expansion:
Explore additional features and factors that could contribute to better predictions, such as player fitness data, team dynamics, weather conditions, and historical player-performance against specific opponents.

3)Real-Time Analytics:
Implement more sophisticated real-time analytics capabilities, allowing users to access dynamic insights during live matches, including instant updates on key events and their impact on predictions.
.

4)User Customization:
Provide users with more extensive customization options for their predictions, allowing them to tailor the model to their specific preferences and priorities.

5)Mobile App Enhancements:
Continuously improve and optimize the mobile app, considering user feedback and incorporating new features that enhance the user experience on mobile devices.

6)Cross-Sport Predictions:
Expand the scope of the predictive model to cover other sports, offering a broader range of predictions and engaging a larger audience interested in different sporting events.

7)Integration with Smart Devices:
Explore integration with smart devices, such as wearables or smart TVs, to provide users with seamless access to predictions and real-time updates.

It's important to keep abreast of technological advancements, user preferences, and industry developments to guide the future evolution of the T20 score prediction ML model project. Regularly seek feedback from users and stakeholders to inform improvements and refinements.

REFERENCES :

[1] Prasad Thorat, Vighnesh Buddhivant, Yash Sahane; Review Paper on Cricket Score Prediction;

[2] Tejinder Singh, Vishal Singla, Parteek Bhatia; - Score and Winning Prediction in Cricket through Data Mining;,

[3] D. Thenmozhi, P. Mirunalini, S. M. Jaisakthi, Srivatsan Vasudevan , Veeramani Kannan V; Moneyball -Data Mining on Cricket Dataset;

https://www.geeksforgeeks.org/ml-linear-regression/

https://towardsdatascience.com/machine-learning-basics-random-forest-regression-be3e1e3bb91a

https://www.geeksforgeeks.org/implementation-of-lasso-regression-from-scratch-using-python/

https://towardsdatascience.com/which-evaluation-metric-should-you-use-in-machine-learning-regression-problems

APPENDIX

SOURCE CODE-

```
# Importing Necessary Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#Importing dataset
ipl_df = pd.read_csv('ipl_data.csv')
print(f"Dataset successfully Imported of Shape : {ipl_df.shape}")
Exploratory Data Analysis
# First 5 Columns Data
ipl_df.head(10)
# Describing the ipl_dfset
ipl_df.describe()
# Information about Each Column
ipl_df.info()
# Number of Unique Values in each column
ipl_df.nunique()
# ipl_df types of all Columns
ipl_df.dtypes
#Wickets Distribution
sns.displot(ipl_df['wickets'],kde=False,bins=10)
plt.title("Wickets Distribution")
plt.show()
#Runs Distribution
sns.displot(ipl_df['total'],kde=False,bins=10)
```

```
plt.title("Runs Distribution")


plt.show()
```

Data Cleaning

Removing Irrelevant Data colunms

 # Names of all columns

```
ipl_df.columns
```

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

```
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()
```

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()
```

Remove First 5 Overs of every match

```
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()
```

Plotting a Correlation Matrix of current data

```
from seaborn import heatmap
heatmap(data=ipl_df.corr(), annot=True)
Data Preprocessing and Encoding
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()
```

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()
```

Model Building

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}")
```

ML Algorithms

```
models = dict()
```

1. Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor
```

```
tree = DecisionTreeRegressor()

# Train Model

tree.fit(train_features, train_labels)

# 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

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)))))
```

Linear Regression

```
from sklearn.linear_model import LinearRegression

linreg = LinearRegression()

# Train Model

linreg.fit(train_features, train_labels)

# 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

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)))))
```

Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor

forest = RandomForestRegressor()

# Train Model

forest.fit(train_features, train_labels)

# 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

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)))))
```

Support Vector Machine

```
!pip install sklearn
```

```
from sklearn.svm import SVR

svm = SVR()

# Train Model

svm.fit(train_features, train_labels)

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

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)))))
```

XGBoost

```
from xgboost import XGBRegressor

xgb = XGBRegressor()

# Train Model

xgb.fit(train_features, train_labels)

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

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)))))
```

KNR

```
from sklearn.neighbors import KNeighborsRegressor

knr = KNeighborsRegressor()

# Train Model

knr.fit(train_features, train_labels)

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

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)))))
```

Best Model

```
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)

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

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curv
e
```

Predictions

```
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]))
```

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')

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')

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')

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')

Test 5

Batting Team : Delhi Daredevils

Bowling Team : Mumbai Indians

Final Score : 110/7

batting_team='Delhi Daredevils'

bowling_team='Mumbai Indians'

score = score_predict(batting_team, bowling_team, overs=18.0, runs=96, wickets=8, runs_last_5=18, wickets_last_5=4)

print(f'Predicted Score : {score} || Actual Score : 110')

Test 6

Batting Team : Kings XI Punjab

Bowling Team : Chennai Super Kings

Final Score : 153/9

batting_team='Kings XI Punjab'

bowling_team='Chennai Super Kings'

score = score_predict(batting_team, bowling_team, overs=18.0, runs=129, wickets=6, runs_last_5=34, wickets_last_5=2)

print(f'Predicted Score : {score} || Actual Score : 153')

Test 7

Batting Team : Sunrisers Hyderabad

Bowling Team : Royal Challengers Banglore

Final Score : 146/10

batting_team='Sunrisers Hyderabad'

bowling_team='Royal Challengers Bangalore'

score = score_predict(batting_team, bowling_team, overs=10.5, runs=67, wickets=3, runs_last_5=29, wickets_last_5=1)

print(f'Predicted Score : {score} || Actual Score : 146')

Export Model

```
import pickle

filename = "ml_model2.pkl"

pickle.dump(forest, open(filename, "wb"))
```

CODE FOR WEBINTERFACE-STREAMLIT

```
#import the libraries

import math

import numpy as np

import pickle

import streamlit as st

#SET PAGE WIDE

st.set_page_config(page_title='T20_Score(IPL)_Predictor',layout="centered")

#Get the ML model

filename='ml_model.pkl'

model = pickle.load(open(filename,'rb'))

#Title of the page with CSS

st.markdown("<h1 style='text-align: center; color: white;'> T20 Score Predictor(ipl) 2023</h1>", unsafe_allow_html=True)
```

```
#Add background image
st.markdown(
        f"""
        <style>
        .stApp {{
            background-image: url("https://4.bp.blogspot.com/-F6aZF5PMwBQ/Wrj5h204qxI/AAAAAAAABao/4QLn48RP3x0P8Ry0CcktxilJqRfv1IfcACLcBGAs/s1600/GURU%2BEDITZ%2Bbackground.jpg");
            background-attachment: fixed;
            background-size: cover
        }}
        </style>
        """,
        unsafe_allow_html=True
    )
#Add description
with st.expander("Description"):
   st.info("""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
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]
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 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)
```

SOURCECODE LINK:

https://colab.research.google.com/drive/1cVJFUFqy6fmdlXL6pN26pMBckbuV-CQZ?usp=sharing

GITHUB LINK:

https://github.com/jayanth3002/T20_score_predictior

VIDEO LINK:

THANKYOU

PREPARED BY-

D.JAYANTH VARMA-21BCE9786

S.SAI BABU-21BCE9232

K.SAI VARDHAN-21BCE9284

VIT-AP UNIVERSITY