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.

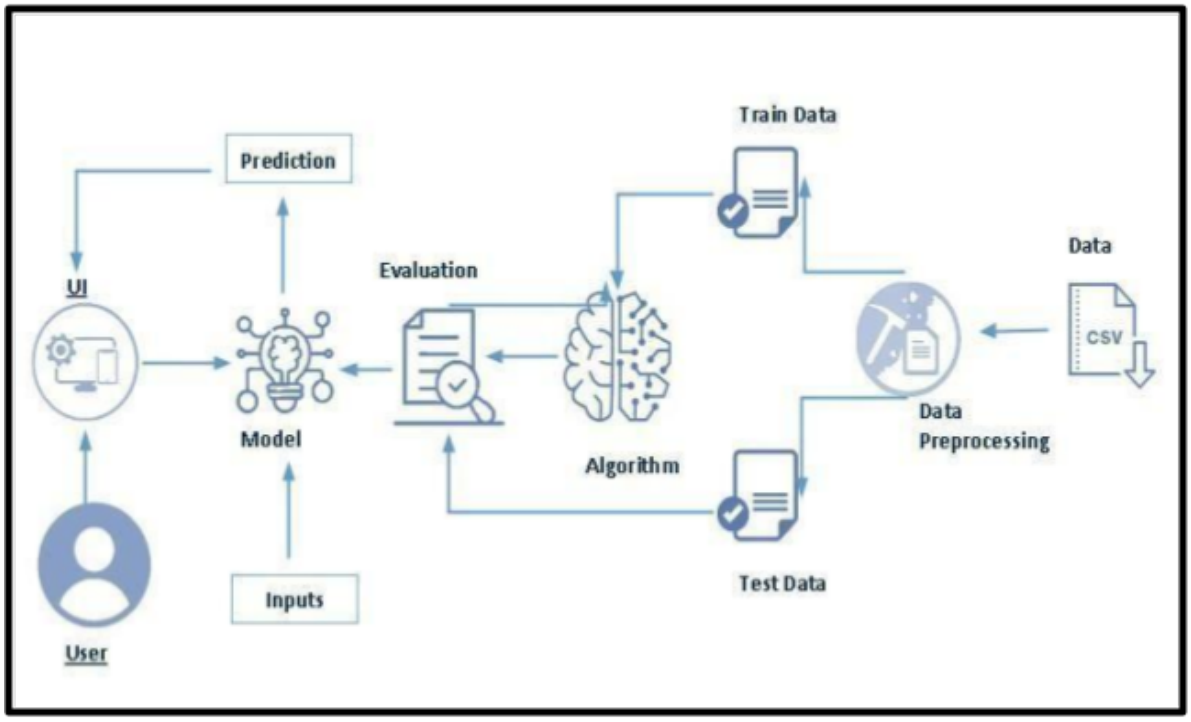
2. **Strategic Decision Making:** Cricket teams, coaches, and analysts can use the predictive model to make informed decisions during matches. This could involve adjusting strategies based on predicted scores, understanding the impact of different factors on the outcome, and optimizing team compositions.

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
  - 0 Specify the business problem
  - Business requirements
  - Literature Survey.
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis
  - 0 Descriptive statistical
  - Visual Analysis
- Model Building
  - Building a model.
  - 0 Training the model.
  - Testing the model
- Model Deployment
  - 0 Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation
  - 0 Record explanation Video for project end to end solution

- Project Documentation-Step by step project development procedure

Prior Knowledge:

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

- VS Code :
  - Refer to the link below to download VS Code.
  - Link : https://code.visualstudio.com/download
- Create Project:
  - Open VS Code.
  - Create a Folder and name it "T20_Score_Predictor" .
- Machine Learning Concepts
  - 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/

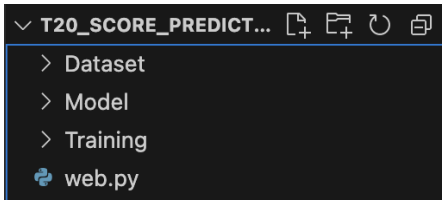
Project Objectives:

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 Streamlit 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.
- The Training folder contains 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

Refer Project Description

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 (Student Will Write)

A literature survey would involve researching and reviewing existing studies, articles, and other publications on the topic of project. The survey would aim to gather information on current systems, their strengths and weaknesses, and any gaps in knowledge that the project could address. The literature survey would also look at the methods and techniques used in previous projects, and any relevant data or findings that could inform the design and implementation of the current project.

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.

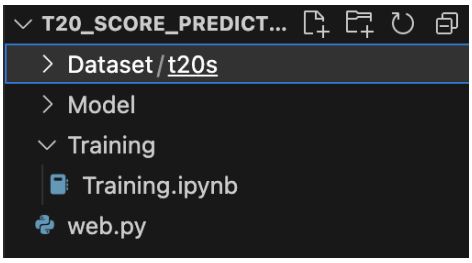
There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

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

CLICK ME

Download only “t20s” 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 1.1: Importing the libraries

Import the necessary libraries as shown in the image.

```
import numpy as np
import pandas as pd
from yaml import safe_load
import yaml
import os
from tqdm import tqdm
```

Activity 1.2: Converting .yaml files to dataframes.

Reading filename :

```
filenames = []
for file in os.listdir('../Dataset/t20s/'):
    filenames.append(os.path.join('../Dataset/t20s/',file))
```

```
filenames[0:5]
```

```
['../Dataset/t20s/1173055.yaml',
 '../Dataset/t20s/951375.yaml',
 '../Dataset/t20s/1188621.yaml',
 '../Dataset/t20s/540173.yaml',
 '../Dataset/t20s/1034829.yaml']
```

Conversion :

```
final_df = pd.DataFrame()
counter = 1
for file in tqdm(filenames):
    try:
        with open(file, 'r') as f:
            df = pd.json_normalize(safe_load(f))
            df['match_id'] = counter
            final_df = final_df.append(df)
    except UnicodeDecodeError:
        print(f"Error processing {file}. Skipping.")
    counter += 1
```

Below shown dataframe should be you output.

```
final_df
```

	innings	meta.data_version	meta.created	meta.revision	info.city	info.dates	info.gender	info.match_type	info.match_type_number	info.outcome.wi
0	[{'1st innings': {'team': 'West Indies', 'deli...	0.9	2020-03-01	1	Canberra	[2020-02-26]	female	T20	853.0	Pak
0	[{'1st innings': {'team': 'India', 'deliveries...	0.9	2016-03-17	2	Bangalore	[2016-03-15]	female	T20	NaN	
0	[{'1st innings': {'team': 'West Indies', 'deli...	0.9	2019-08-06	1	Lauderhill	[2019-08-03]	male	T20	842.0	
	[{'1st									

Milestone 3: Exploratory Data Analysis

Activity 1: Dropping unnecessary columns

```
final_df.drop(columns=[
    'meta.data_version',
    'meta.created',
    'meta.revision',
    'info.outcome.bowl_out',
    'info.bowl_out',
    'info.supersubs.South Africa',
    'info.supersubs.New Zealand',
    'info.outcome.eliminator',
    'info.outcome.result',
    'info.outcome.method',
    'info.neutral_venue',
    'info.match_type_number',
    'info.outcome.by.runs',
    'info.outcome.by.wickets'
],inplace=True)
```

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

- We'll be predicting the scores for men's t20 only, because there is no sufficient data to train the machine learning model for women's t20 also.

```
final_df['info.gender'].value_counts()
```

```
male      966
female    466
Name: info.gender, dtype: int64
```

```
final_df = final_df[final_df['info.gender'] == 'male']
final_df.drop(columns=['info.gender'],inplace=True)
final_df
```

Now we have dropped the gender column and reduced the number of rows.

- The given dataset also contains some 50 over matches.

```
final_df['info.match_type'].value_counts()
```

```
T20    966
Name: info.match_type, dtype: int64
```

```
final_df['info.overs'].value_counts()
```

```
20    963
50      3
Name: info.overs, dtype: int64
```

We've found three 50 over matches. Let's remove them.

```
final_df = final_df[final_df['info.overs'] == 20]
final_df.drop(columns=['info.overs','info.match_type'],inplace=True)
final_df
```

As of now, we've achieved level 1 of EDA. Now we'll be going to extract the data of all the matches.

Let's save the preprocessed data.

```
import pickle
pickle.dump(final_df,open('dataset_level1.pkl','wb'))
```

Next step will be performing the EDA in 'dataset_level1'.

```
matches = pickle.load(open('dataset_level1.pkl','rb'))
matches.iloc[0]['innings'][0]['1st innings']['deliveries']
```

The above code is to just extract a single match with data for each ball bowled.

Now we'll create a dataframe with required columns using below code.

```
count = 1
delivery_df = pd.DataFrame()
for index, row in matches.iterrows():
    if count in [75,108,150,180,268,360,443,458,584,748,982,1052,1111,1226,1345]:
        count+=1
        continue
    count+=1
    ball_of_match = []
    batsman = []
    bowler = []
    runs = []
    player_of_dismissed = []
    teams = []
    batting_team = []
    match_id = []
    city = []
    venue = []
    for ball in row['innings'][0]['1st innings']['deliveries']:
        for key in ball.keys():
            match_id.append(count)
            batting_team.append(row['innings'][0]['1st innings']['team'])
            teams.append(row['info.teams'])
            ball_of_match.append(key)
            batsman.append(ball[key]['batsman'])
            bowler.append(ball[key]['bowler'])
            runs.append(ball[key]['runs']['total'])
            city.append(row['info.city'])
            venue.append(row['info.venue'])
            try:
                player_of_dismissed.append(ball[key]['wicket']['player_out'])
            except:
                player_of_dismissed.append('0')
    loop_df = pd.DataFrame({
            'match_id':match_id,
            'teams':teams,
            'batting_team':batting_team,
            'ball':ball_of_match,
            'batsman':batsman,
            'bowler':bowler,
            'runs':runs,
            'player_dismissed':player_of_dismissed,
            'city':city,
            'venue':venue
        })
    delivery_df = delivery_df.append(loop_df)
```

You'll get some 1.15 Lakh rows after extracting it.
Now, Let's extract the bowling team from the teams column and drop the teams column.

```
def bowl(row):
    for team in row['teams']:
        if team != row['batting_team']:
            return team
```

```
delivery_df['bowling_team'] = delivery_df.apply(bowl,axis=1)
```

```
delivery_df
```

Let's remove the unbalanced data i.e teams which played less no.of matches.

```
delivery_df['batting_team'].value_counts()
```

```
teams = [
    'Australia',
    'India',
    'Bangladesh',
    'New Zealand',
    'South Africa',
    'England',
    'West Indies',
    'Afghanistan',
    'Pakistan',
    'Sri Lanka'
]
```

```
delivery_df = delivery_df[delivery_df['batting_team'].isin(teams)]
delivery_df = delivery_df[delivery_df['bowling_team'].isin(teams)]
```

```
delivery_df.drop(columns = ['teams'], inplace=True)
```

We've reduced the no.of rows by eliminating unbalanced data.

The following data is required for our model.

```
output = delivery_df[['match_id','batting_team','bowling_team','ball','runs','player_dismissed','city','venue']]
```

Let's save the dataset as level 2.

```
pickle.dump(output,open('dataset_level2.pkl','wb'))
```

Activity 2: Feature Extraction

- Loading the data

```
df = pickle.load(open('dataset_level2.pkl','rb'))
```

- Looking for null values

```
df.isnull().sum()
```

```
match_id               0
batting_team           0
bowling_team           0
ball                   0
runs                   0
player_dismissed       0
city                8545
venue                  0
dtype: int64
```

- Extracting city names using venue column and dropping venue column

```
df[df['city'].isnull()]['venue'].value_counts()
```

```
Dubai International Cricket Stadium          2966
Pallekele International Cricket Stadium      2066
Melbourne Cricket Ground                     1453
Sydney Cricket Ground                         749
Adelaide Oval                                 498
Harare Sports Club                            372
Sharjah Cricket Stadium                       249
Sylhet International Cricket Stadium          128
Carrara Oval                                   64
Name: venue, dtype: int64
```

```
cities = np.where(df['city'].isnull(), df['venue'].str.split().apply(lambda x:x[0]), df['city'])
```

```
df['city'] = cities
```

```
df.isnull().sum()
```

```
match_id           0
batting_team       0
bowling_team       0
ball               0
runs               0
player_dismissed   0
city               0
venue              0
dtype: int64
```

```
df.drop(columns=['venue'],inplace=True)
```

- Filtering the cities based on the number of balls thrown in each city. If the no.of matches played in each stadium is less than 5 i.e 600 balls, we'll remove that city.

```
eligible_cities = df['city'].value_counts()[df['city'].value_counts()>600].index.tolist()
df = df[df['city'].isin(eligible_cities)]
df
```

- Creating a new column with the current score. Let's write the code to extract the current score.

```
df['current_score'] = df.groupby('match_id').cumsum()['runs']
df
```

- Now let's create two more columns, namely 'over' and 'ball_no'.

```
df['over'] = df['ball'].apply(lambda x:str(x).split(".")[0])
df['ball_no'] = df['ball'].apply(lambda x:str(x).split(".")[1])
```

- Similarly, we'll write the code for 'balls_bowled' and 'balls_left'.

```
df['balls_bowled'] = (df['over'].astype('int')*6) + df['ball_no'].astype('int')
```

```
df['balls_left'] = 120 - df['balls_bowled']
df['balls_left'] = df['balls_left'].apply(lambda x:0 if x<0 else x)
df
```

- Again the same thing for 'player_dismissed','wickets_left' and current run rate('crr').

```
df['player_dismissed'] = df['player_dismissed'].apply(lambda x:0 if x=='0' else 1)
df['player_dismissed'] = df['player_dismissed'].astype('int')
df['player_dismissed'] = df.groupby('match_id').cumsum()['player_dismissed']
df['wickets_left'] = 10 - df['player_dismissed']
```

```
df['crr'] = round((df['current_score']*6)/df['balls_bowled'],2)
df
```

- Extracting the runs in the last 5 overs of every match and adding it as a new column.

```
groups = df.groupby('match_id')

match_ids = df['match_id'].unique()
last_five = []
for id in match_ids:
    last_five.extend(groups.get_group(id).rolling(window=30).sum()['runs'].values.tolist())
```

```
df['last_five'] = last_five
df
```

- Selecting only required features from the dataframe.

  final_df = final_df[['batting_team', 'bowling_team', 'city', 'current_score', 'balls_left', 'wickets_left', 'crr', 'last_five', 'runs_x']]

- Checking for null values in the final dataframe and drop them.

```
final_df.isnull().sum()
```

```
batting_team          0
bowling_team          0
city                  0
current_score         0
balls_left            0
wickets_left          0
crr                   0
last_five         12111
runs_x                0
dtype: int64
```

```
final_df.dropna(inplace=True)
```

- ```
  final_df = final_df.sample(final_df.shape[0])
  ```
- Let's take a look at the final data frame which is ready for training.

```
final_df
```

	batting_team	bowling_team	city	current_score	balls_left	wickets_left	crr	last_five	runs_x
28419	Sri Lanka	India	Mumbai	135	0	3	6.75	38.0	135
10553	Sri Lanka	Australia	Colombo	67	55	6	6.18	22.0	128
44136	India	England	Colombo	119	26	8	7.60	38.0	170
47172	England	Australia	Cape Town	48	74	7	6.26	31.0	135
31868	South Africa	Pakistan	Dubai	47	80	9	7.05	29.0	150
...	...	...	...	...	...	...	...	...	...
43179	New Zealand	Sri Lanka	Auckland	57	55	5	5.26	25.0	179
3698	New Zealand	Sri Lanka	Colombo	117	20	6	7.02	32.0	141
43100	New Zealand	Pakistan	Wellington	176	10	4	9.60	53.0	196
36956	South Africa	England	Cape Town	53	74	7	6.91	30.0	154
20635	New Zealand	India	Johannesburg	90	47	6	7.40	28.0	190

38751 rows × 9 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.
```

```
X = final_df.drop(columns=['runs_x'])
y = final_df['runs_x']
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=1)
```

```
X_train
```

	batting_team	bowling_team	city	current_score	balls_left	wickets_left	crr	last_five
16824	New Zealand	Pakistan	Chandigarh	99	50	8	8.49	40.0
31107	India	New Zealand	Wellington	60	70	7	7.20	31.0
23121	Australia	South Africa	Johannesburg	75	47	6	6.16	42.0
32571	West Indies	India	Kolkata	109	0	2	5.45	46.0
42232	India	South Africa	Durban	141	4	6	7.29	45.0
...	...	...	...	...	...	...	...	...
8430	Sri Lanka	England	Southampton	79	60	7	7.90	42.0
42650	New Zealand	India	Auckland	46	80	7	6.90	42.0
12171	Pakistan	Australia	Mirpur	95	54	8	8.64	59.0
3571	New Zealand	Australia	Sydney	90	23	4	5.57	33.0
41922	South Africa	New Zealand	Johannesburg	83	65	10	9.05	48.0

31000 rows × 8 columns

Activity 4: Importing required packages and Encoding.

- Importing required packaged

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.metrics import r2_score,mean_absolute_error
```

- One hot encoding the data using the Column transfer method.

```
trf = ColumnTransformer([
    ('trf',OneHotEncoder(sparse=False,drop='first'),['batting_team','bowling_team','city'])
]
,remainder='passthrough')
```

Milestone 4: Model Training

In this step we'll select models and train them, step by step.

We'll use 3 machine learning algos and find the best one out.

Model 1: Linear Regression

Creating pipeline

```
pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),
    ('step3',LinearRegression())
])
```

Training and calculating the accuracy

```
pipe.fit(X_train,y_train)
y_pred = pipe.predict(X_test)
print(r2_score(y_test,y_pred))
print(mean_absolute_error(y_test,y_pred))
```

```
0.6885977930742969
13.160578311496517
```

Accuracy is near 68%

Model 2: Random Forest Regressor

Creating Pipeline

```
pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),
    ('step3',RandomForestRegressor())
])
```

Training and calculating the accuracy

```
pipe.fit(X_train,y_train)
y_pred = pipe.predict(X_test)
print(r2_score(y_test,y_pred))
print(mean_absolute_error(y_test,y_pred))
```

```
0.9767823737527738
2.1144844106136844
```

Accuracy is somewhere around 97%

Model 3: XGBRegressor

Creating Pipeline

```
pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),
    ('step3',XGBRegressor(n_estimators=1000,learning_rate=0.2,max_depth=12,random_state=1))
])
```

Training and calculating the accuracy

```
pipe.fit(X_train,y_train)
y_pred = pipe.predict(X_test)
print(r2_score(y_test,y_pred))
print(mean_absolute_error(y_test,y_pred))
```

```
0.9863469919711688
1.6940391234406624
```

Accuracy is somewhere around 98%.

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 XGBRegressor is performing well out of all the models.

```
pickle.dump(pipe,open('../Model/pipe.pkl', 'wb'))
```

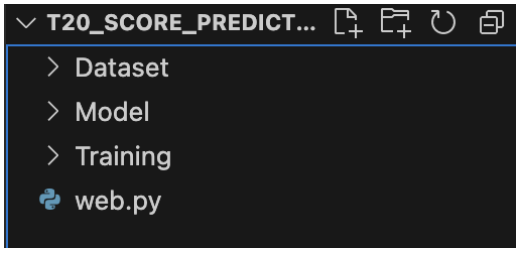
Activity 2: Integrate with Web Framework

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 streamlit as st
import pickle
import pandas as pd
```

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/cities names and loading the pipeline:

```
pipe = pickle.load(open('./Model/pipe.pkl','rb'))

teams = ['Australia',
 'India',
 'Bangladesh',
 'New Zealand',
 'South Africa',
 'England',
 'West Indies',
 'Afghanistan',
 'Pakistan',
 'Sri Lanka']
```

```
cities = ['Colombo',
 'Mirpur',
 'Johannesburg',
 'Dubai',
 'Auckland',
 'Cape Town',
 'London',
 'Pallekele',
 'Barbados',
 'Sydney',
 'Melbourne',
 'Durban',
 'St Lucia',
 'Wellington',
 'Lauderhill',
 'Hamilton',
 'Centurion',
 'Manchester',
 'Abu Dhabi',
 'Mumbai',
 'Nottingham',
 'Southampton',
 'Mount Maunganui',
 'Chittagong',
 'Kolkata',
 'Lahore',
 'Delhi',
 'Nagpur',
 'Chandigarh',
 'Adelaide',
 'Bangalore',
 'St Kitts',
 'Cardiff',
 'Christchurch',
 'Trinidad']
```

Activity 2.3: Accepting the input from user and prediction

```
st.title('Cricket Score Predictor')

col1, col2 = st.columns(2)

with col1:
    batting_team = st.selectbox('Select batting team',sorted(teams))
with col2:
    bowling_team = st.selectbox('Select bowling team', sorted(teams))

city = st.selectbox('Select city',sorted(cities))

col3,col4,col5 = st.columns(3)

with col3:
    current_score = st.number_input('Current Score')
with col4:
    overs = st.number_input('Overs done(works for over>5)')
with col5:
    wickets = st.number_input('Wickets out')

last_five = st.number_input('Runs scored in last 5 overs')

if st.button('Predict Score'):
    balls_left = 120 - (overs*6)
    wickets_left = 10 -wickets
    crr = current_score/overs

    input_df = pd.DataFrame(
    {'batting_team': [batting_team],
     'bowling_team': [bowling_team],
     'city':city, 'current_score': [current_score],
     'balls_left': [balls_left],
     'wickets_left': [wickets],
     'crr': [crr],
     'last_five': [last_five]
     })
    result = pipe.predict(input_df)
    st.header("Predicted Score - " + str(int(result[0])))
```

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

- “streamlit run web.py”

```
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.1.11:8501

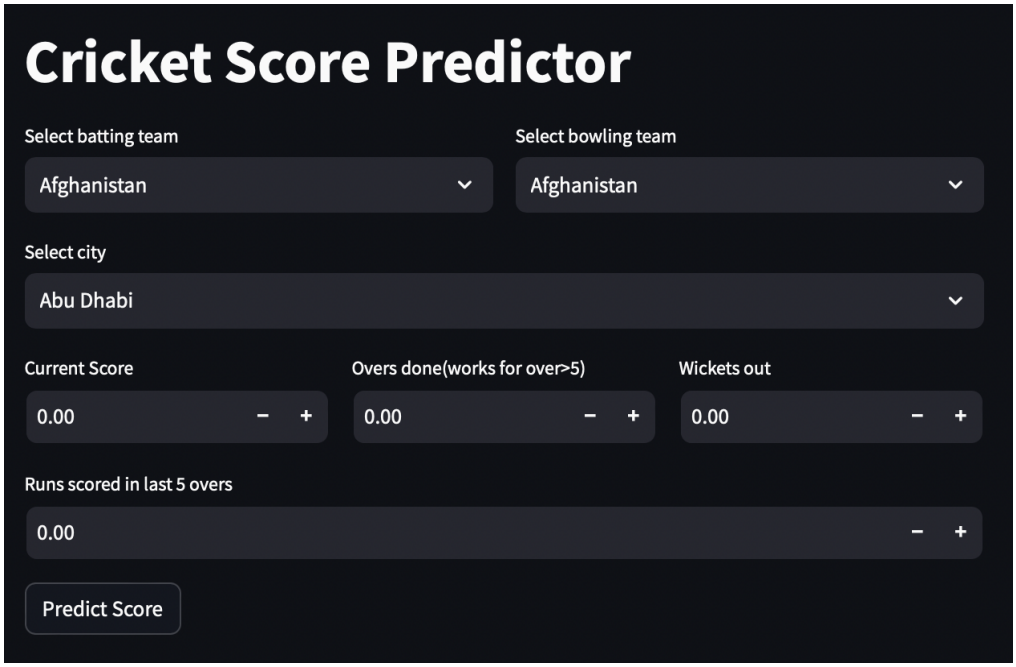
For better performance, install the Watchdog module:

$ xcode-select --install
$ pip install watchdog
```

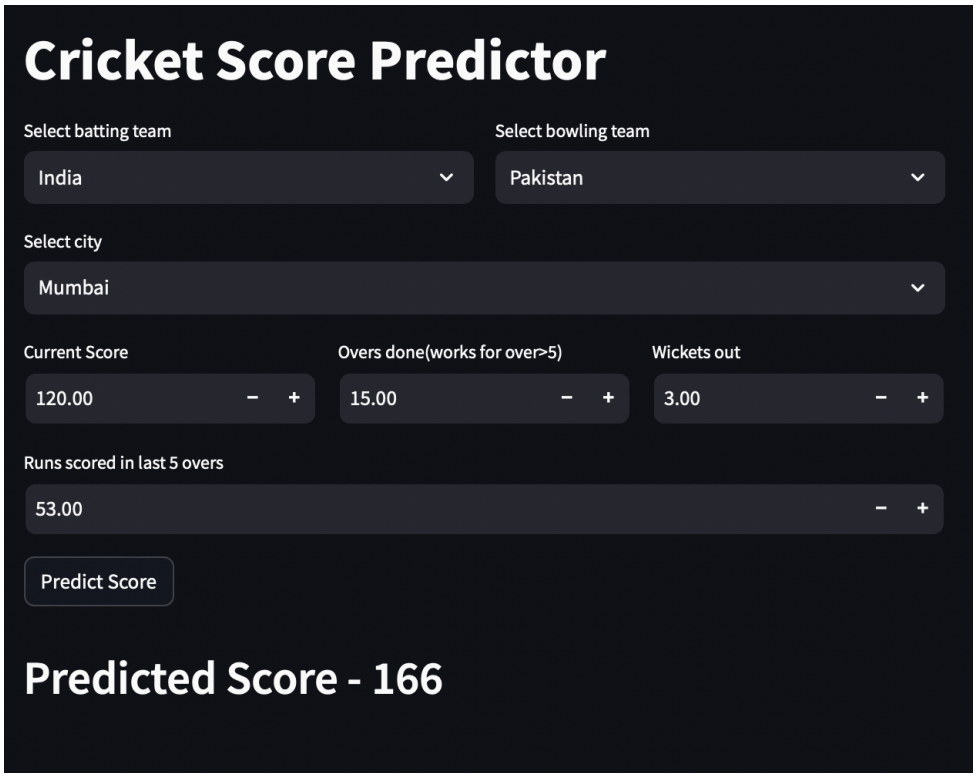
On successful execution you'll get to see this in your terminal.

HOW DOES YOUR WEBSITE LOOK LIKE ?

- Before entering the data :



- After entering the data :



<u>Milestone 6: Project Demonstration & Documentation</u>

Below mentioned deliverables to be submitted along with other deliverables.

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

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

Create a document as per the template provided.