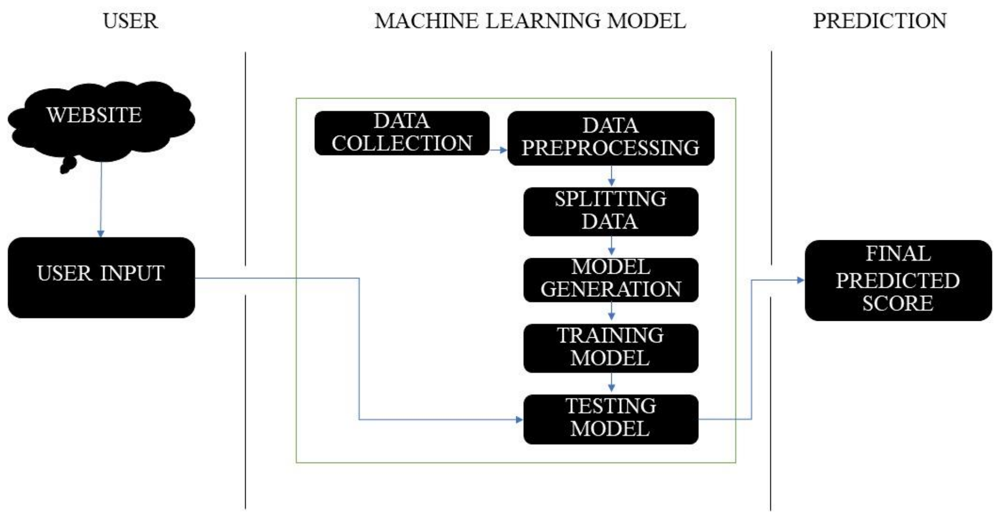
T20 Totalitarian: Mastering Score Predictions

Introduction:

In the fast-paced world of modern cricket, Twenty20 (T20) has emerged as the format of choice for both players and fans. With its shorter duration, high-octane action, and global appeal, T20 cricket has revolutionized the sport and captivated audiences worldwide. However, beyond the excitement on the field, another phenomenon has gained immense traction in recent years - the art and science of predicting T20 cricket scores.

The unpredictability and thrill associated with T20 matches make score prediction an enticing endeavour for cricket enthusiasts, data analysts, and sports enthusiasts alike. It goes beyond mere speculation; it represents an intersection of advanced statistical modelling, machine learning, and the rich history of the sport. This project, "T20 Totalitarian: Mastering Score Predictions," delves into this exciting domain, aiming to explore the key facets of T20 score predictions, the methodologies involved, and their significance in enhancing the viewing experience and contributing to cricket's ever-evolving strategies.

Technical Architecture:



Prerequisites:

Visual Studio Code is a free and open-source distribution of the Python programming language for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Pip is an open-source, cross-platform, package management system.

For this project we will be using Visual Studio Code.

1. To build Machine learning models you must require the following packages

● **Numpy:**

o It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations

● **Scikit-learn:**

o It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy

● **Streamlit:**

Web framework used for building Web applications

● **Python packages:**

o open command prompt as administrator

o Type "pip install numpy" and click enter.

o Type "pip install pandas" and click enter.

o Type "pip install scikit-learn" and click enter.

o Type "pip install tensorflow==2.3.2" and click enter.

o Type "pip install keras==2.3.1" and click enter.

o Type "pip install streamlit" and click enter.

● **Machin Learning Concepts:**

o **Linear Regression:**

Linear regression is a simple and interpretable model used for predicting a numeric target variable. It assumes a linear relationship between the input features and the target variable. In your project, you've used linear regression to build a linear model for score prediction.

o **Random Forest Regressor:**

Random Forest is an ensemble learning technique that combines multiple decision trees to make predictions. Random Forest Regressor is used for regression tasks and is robust, capable of handling both numerical and categorical data. It often provides better predictive performance than a single decision tree.

o **XGBoost Regressor:**

XGBoost (Extreme Gradient Boosting) is another ensemble method that is highly effective for regression tasks. It is known for its speed and accuracy. XGBoost Regressor builds an ensemble of decision trees, continuously improving the model's predictions by minimizing the residual errors.

o **Streamlit**:

Streamlit is a lightweight and user-friendly web application framework that allows you to turn data scripts into shareable web apps. It's designed to be simple to use and is particularly well-suited for data scientists and machine learning engineers.

Project Objectives:

By the end of this project, you will:

● Know fundamental concepts and techniques of Machine Learning like Linear Regression, Random Forest Regressor and XGBoost Regressor.

● Gain a broad understanding of tabular data.

● 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 Flow:

● The user interacts with the UI (User Interface) to choose the input.

● The given input is analysed by the model which is integrated with flask application.

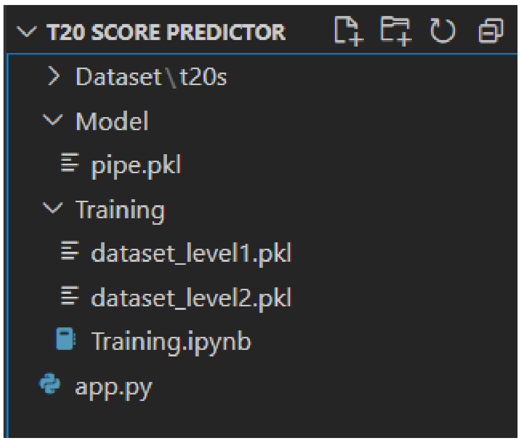
● XGBoost Regressor model analyses the input, then prediction is showcased on the Streamlit UI.

To accomplish this, we have to complete all the activities and tasks listed below

- o **Data Collection**
  - o Create Train and Test Folders.
- o **Data Preprocessing**
  - o Data Cleaning and Handling Missing Values
  - o Feature Engineering
  - o Data Splitting for Model Training and Testing
- o **Model Building**
  - o Import the model building Libraries
  - o Data loading
  - o Data splitting
  - o Feature selection and engineering
  - o Model selection
  - o Model initialization
  - o Training the models
  - o Model evaluation
  - o Selecting the best model
  - o Save the model
- o **Application Building**
  - o Create an HTML file
  - o Building Python code

Project Structure:

Create a Project folder which contains files as shown below.



```
T20 SCORE PREDICTOR
  > Dataset\t20s
  v Model
      pipe.pkl
  v Training
      dataset_level1.pkl
      dataset_level2.pkl
      Training.ipynb
    app.py
```

- The Dataset folder contains another folder t20s which contains all the yaml files used for our model.
- We are building a Streamlit Application that needs a python script app.py for server-side scripting.
- We need the model which is saved and the saved model in this content is a pipe.pkl
- Training folder contains the Training.ipynb file to create and train the model.

Milestone 1: Data Collection

Collect the yaml files of the t20s data and the organize into the Dataset folder by creating a t20s folder in it as shown in the project structure. In this project we have collected only t20s data and saved in the respective directory with its name.

Download the Dataset-
https://www.kaggle.com/datasets/veeralakrishna/cricsheet-a-retrosheet-for-cricket?select=t20s

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

Milestone 2: Image Preprocessing

In the data preprocessing step of this project, you'll perform several essential tasks to prepare the raw 't20s' cricket match dataset for machine learning model training. Data preprocessing is a critical phase that ensures the dataset is clean, structured, and suitable for predictive modelling.

Activity 1: Importing the necessary libraries.

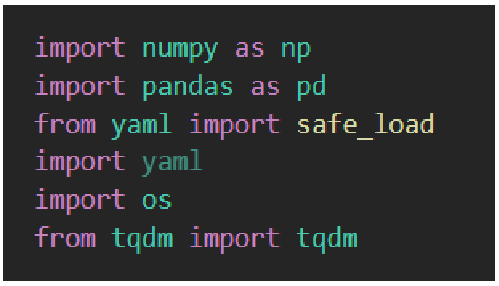
NumPy is a library for numerical computations in Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

Pandas is a powerful library for data manipulation and analysis. It provides data structures like DataFrames and Series, making it easier to work with structured data.

YAML is a human-readable data serialization format. The yaml module is used for parsing and working with YAML files. safe_load is a function that safely loads YAML data, helping prevent code execution from untrusted sources.

The os module provides a way to interact with the operating system, allowing you to perform various file and directory operations.

TQDM is a library that provides a progress bar for loops and iterations. It's useful for tracking the progress of tasks that might take a while to complete.

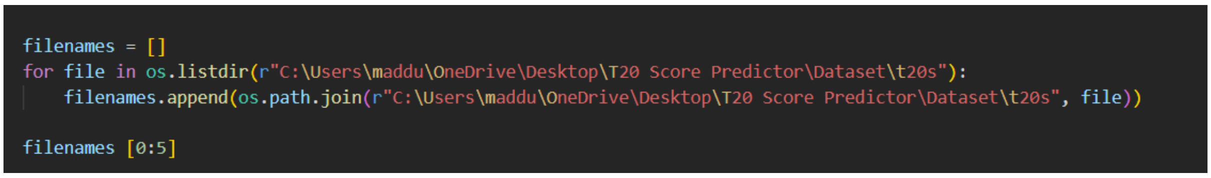


```python
import numpy as np
import pandas as pd
from yaml import safe_load
import yaml
import os
from tqdm import tqdm
```

Activity 2: Converting .yaml files to dataframes.

Reading the filename:

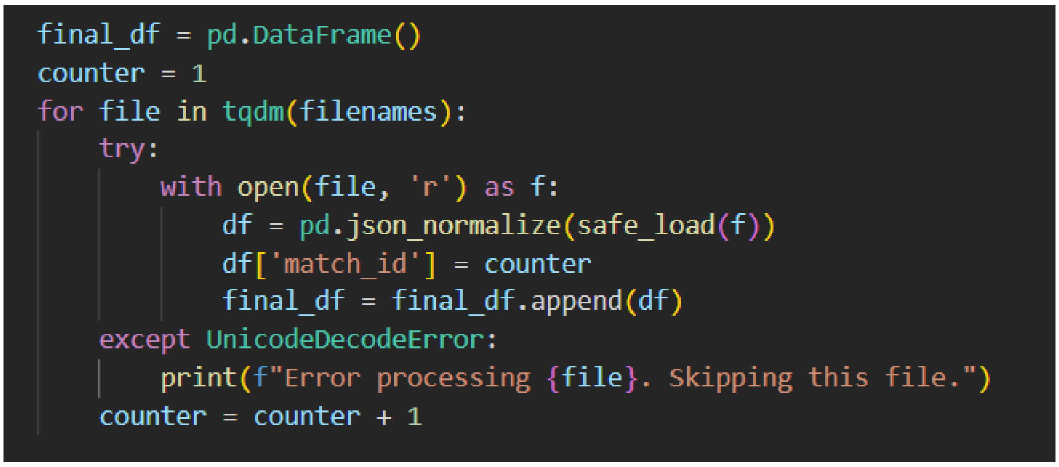
Creates an empty list called "filenames" and then iterate through the files in the specified directory path where the yaml files are stored. For each file in that directory, append the full file path to the "filenames" list.



```python
filenames = []
for file in os.listdir(r"C:\Users\maddu\OneDrive\Desktop\T20 Score Predictor\Dataset\t20s"):
    filenames.append(os.path.join(r"C:\Users\maddu\OneDrive\Desktop\T20 Score Predictor\Dataset\t20s", file))

filenames [0:5]
```

Conversion:

Create a new DataFrame named ‘final_df’ which is used to accumulate the data from multiple files. Then iterate through the list of filenames in the ‘filenames’ list. For each file open and load the data.

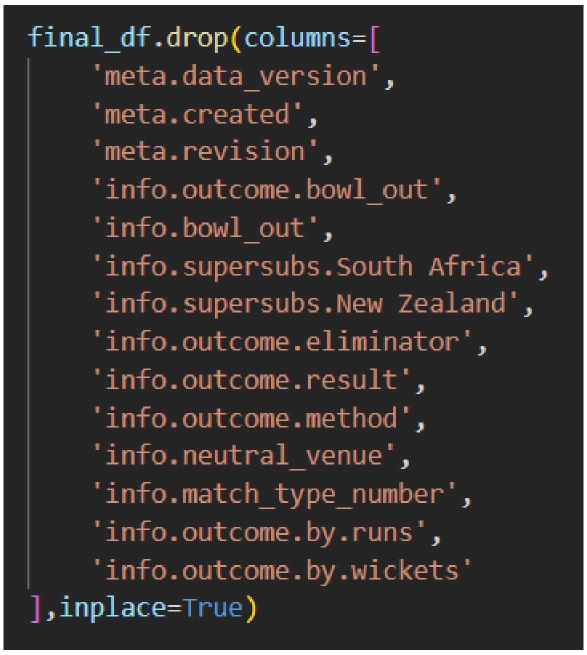


```
final_df = pd.DataFrame()
counter = 1
for file in tqdm(filenames):
    try:
        with open(file, 'r') as f:
            df = pd.json_normalize(safe_load(f))
            df['match_id'] = counter
            final_df = final_df.append(df)
    except UnicodeDecodeError:
        print(f"Error processing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

Milestone 3: Exploratory Data Analysis

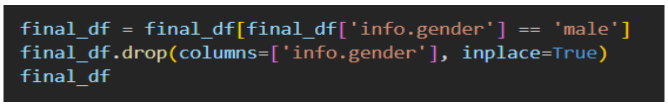
Activity 1: Dropping unnecessary columns.

Let us drop the columns which are unnecessary for our prediction.



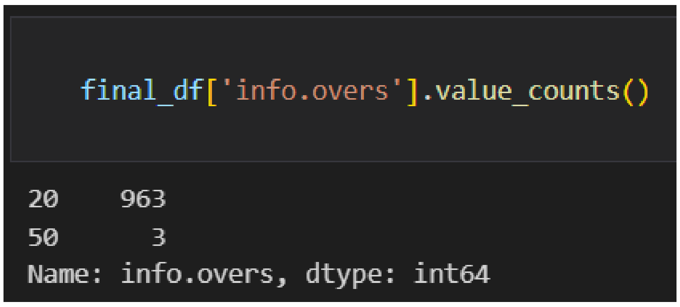
```
final_df.drop(columns=[
    'meta.data_version',
    'meta.created',
    'meta.revision',
    'info.outcome.bowl_out',
    'info.bowl_out',
    'info.supersubs.South Africa',
    'info.supersubs.New Zealand',
    'info.outcome.eliminator',
    'info.outcome.result',
    'info.outcome.method',
  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

We’ll be predicting the scores for men’s t20 only. So let us drop the data of women’s cricket and drop the gender column.



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

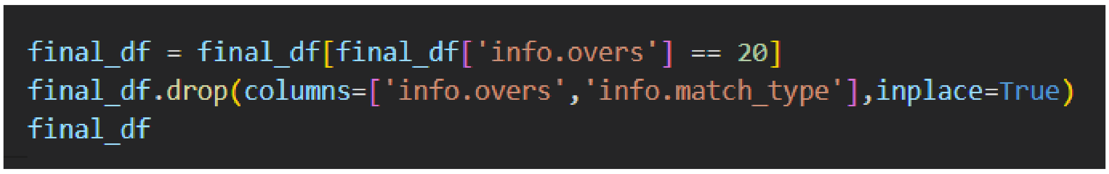
The given dataset also contains some 50 over matches.



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

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

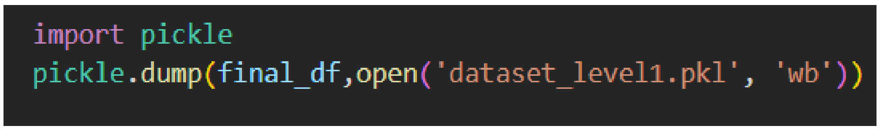
We’ve found three 50 over matches. Let’s remove them.



```
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 Exploratory Data Analysis. Now we’ll be going to extract the data of all the matches.

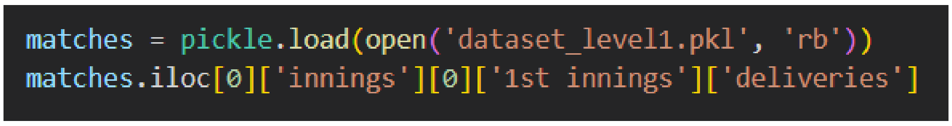
Let’s save the pre-processed data.



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

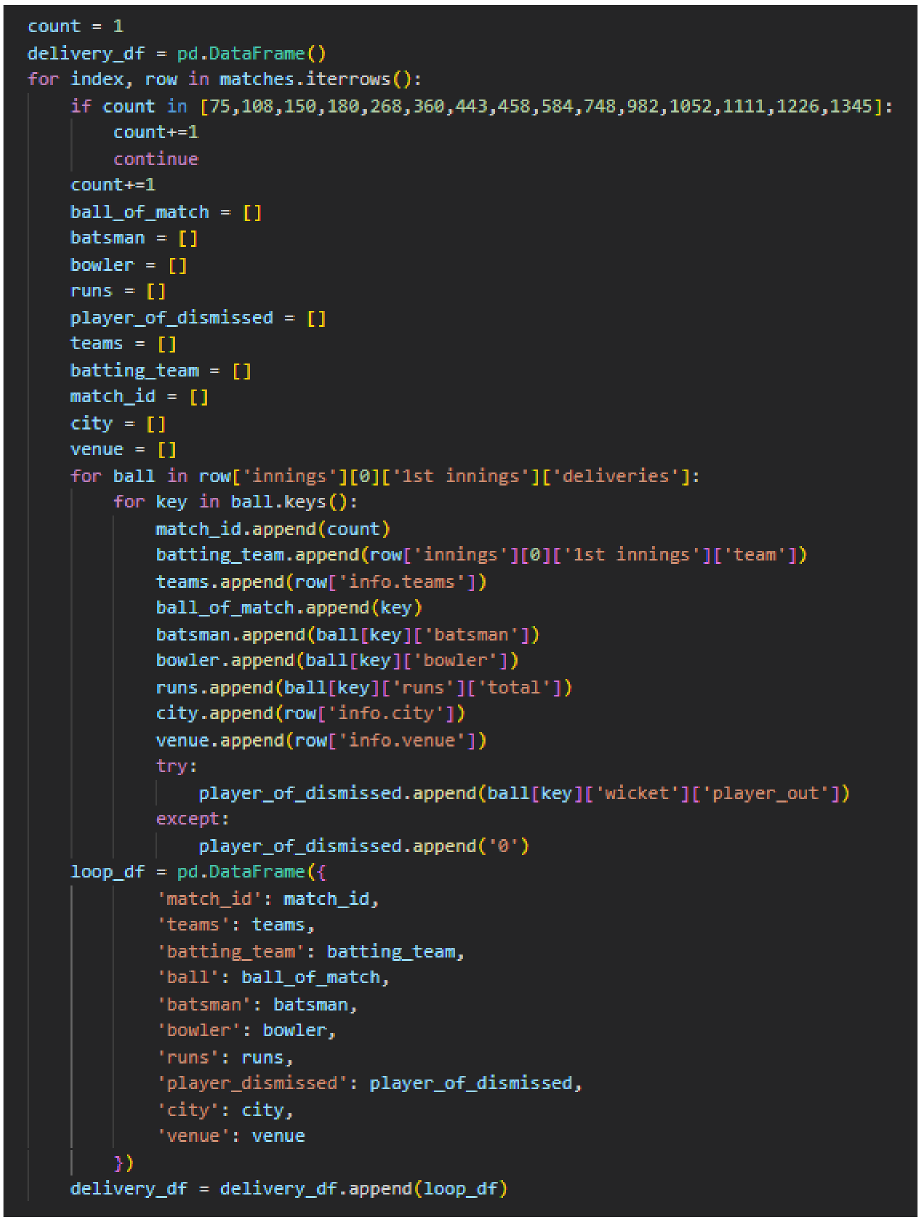
Next step is performing the Exploratory Data Analysis in ‘dataset_level1’.

Extract the data of a single match for each ball bowled.



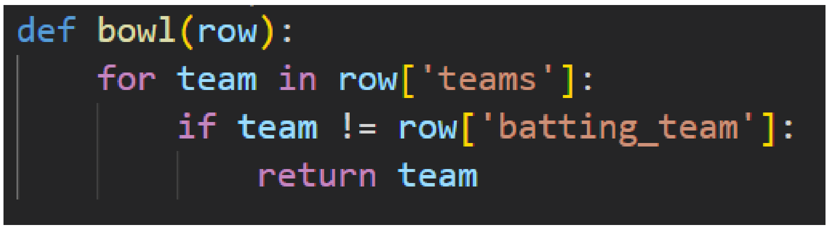
```
matches = pickle.load(open('dataset_level1.pkl', 'rb'))
matches.iloc[0]['innings'][0]['1st innings']['deliveries']
```

Create a dataframe with required columns.

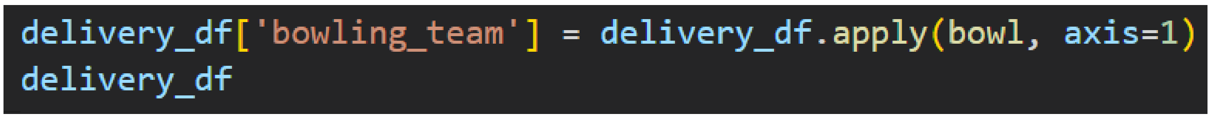


```python
count = 1
delivery_df = pd.DataFrame()
for index, row in matches.iterrows():
    if count in [75,108,150,180,268,360,443,458,584,748,982,1052,1111,1226,1345]:
        count+=1
        continue
    count+=1
    ball_of_match = []
    batsman = []
    bowler = []
    runs = []
    player_of_dismissed = []
    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)
```

Extract the bowling team from the teams column and drop the teams column.



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



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

Remove the unbalanced data that is teams which played less number of matches.

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

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

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

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

The following data is required for our model.

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

Save the dataset as level 2.

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

Activity 2: Feature Extraction

Load the data.

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

Check if there are any null values.

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

```
match_id            0
batting_team        0
bowling_team        0
ball                0
runs                0
player_dismissed    0
city             8548
venue               0
dtype: int64
```

Extract city names using venue column and drop the venue column.

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

```
Dubai International Cricket Stadium        2969
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(column=['vanue'],inplace=True)
```

Filter the cities based on the number of balls thrown in each city. If the number of matches played in each stadium is less than 5 that is less than 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
```

Create a new column ‘current_score’.

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

Create two columns named ‘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, create two more columns named '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)
```

Similarly creating ‘player_dissmissed’, ‘wickets_left’, and ‘crr’ (current run rate).

```
df['player_dismissed'] = df['player_dismissed'].apply(lambda x: 1 if x != '0' else 0)
```

```
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'] = (df['current_score']*6) / df['balls_bowled']
```

Extract the runs in the last 5 overs of every match and adding it as the 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
```

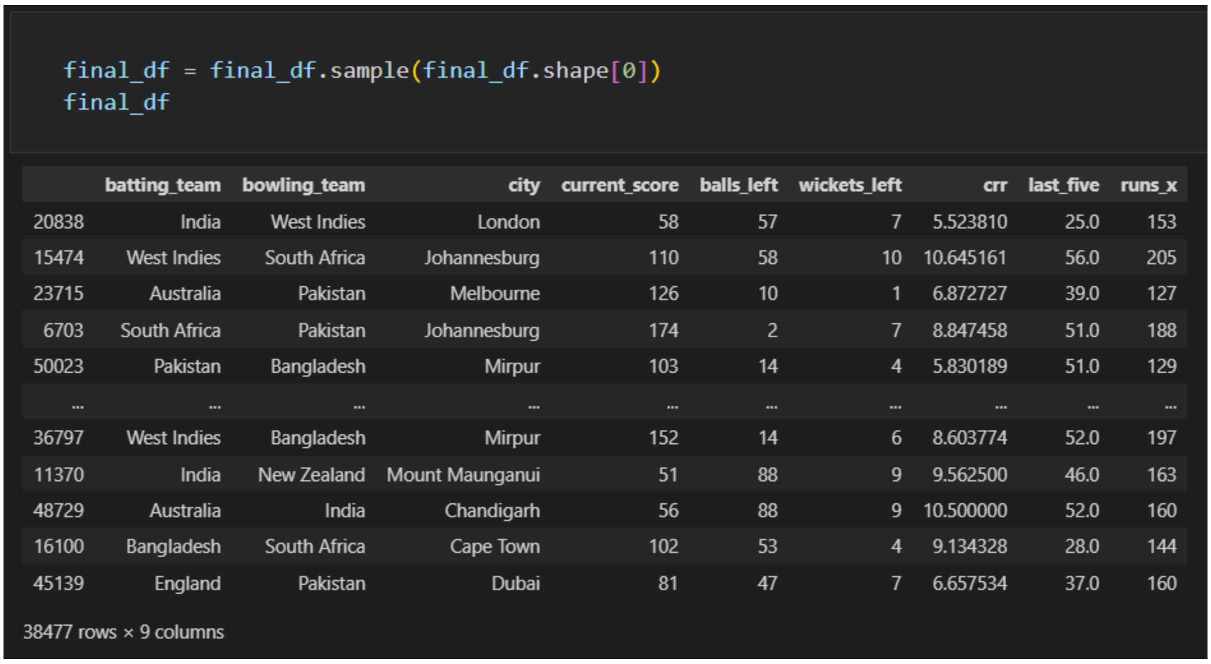
Selecting only required features of the dataframe.

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

Check for null values in the final dataframe and drop them.

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

Let’s look at the final data frame which is ready for training.

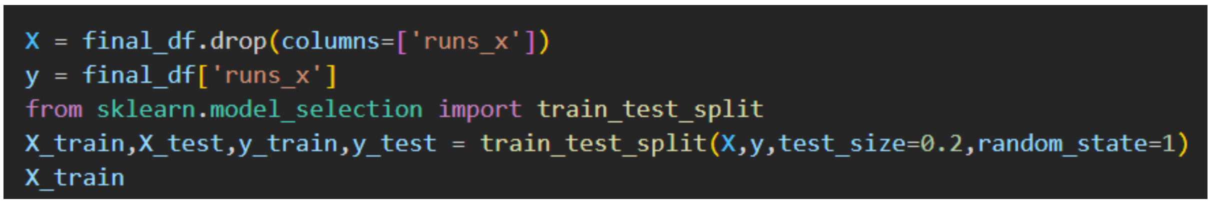


```
final_df = final_df.sample(final_df.shape[0])
final_df
```

```
      batting_team  bowling_team             city  current_score  balls_left  wickets_left        crr  last_five  runs_x
20838        India   West Indies           London             58          57             7   5.523810       25.0     153
15474  West Indies  South Africa     Johannesburg            110          58            10  10.645161       56.0     205
23715    Australia      Pakistan        Melbourne            126          10             1   6.872727       39.0     127
 6703 South Africa      Pakistan     Johannesburg            174           2             7   8.847458       51.0     188
50023     Pakistan    Bangladesh           Mirpur            103          14             4   5.830189       51.0     129
  ...          ...           ...              ...            ...         ...           ...        ...        ...     ...
36797  West Indies    Bangladesh           Mirpur            152          14             6   8.603774       52.0     197
11370        India   New Zealand  Mount Maunganui             51          88             9   9.562500       46.0     163
48729    Australia         India       Chandigarh             56          88             9  10.500000       52.0     160
16100   Bangladesh  South Africa        Cape Town            102          53             4   9.134328       28.0     144
45139      England      Pakistan            Dubai             81          47             7   6.657534       37.0     160

38477 rows × 9 columns
```

Activity 3: Splitting data into train and test sets.

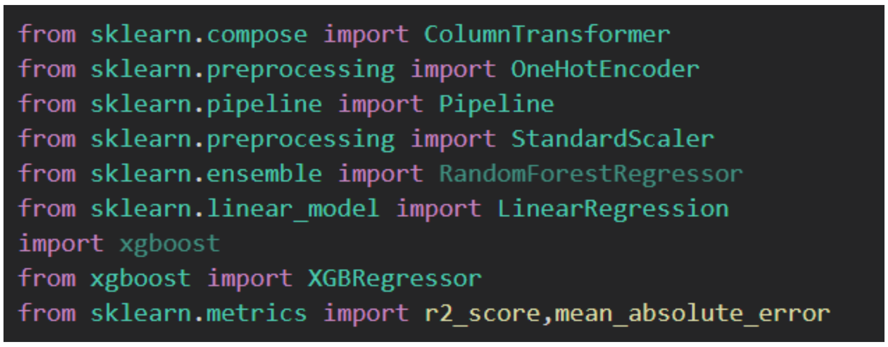
Now we need to split the Dataset into train and test sets. This split will be done in 80:20 ratio – train : test respectively.



```
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
```

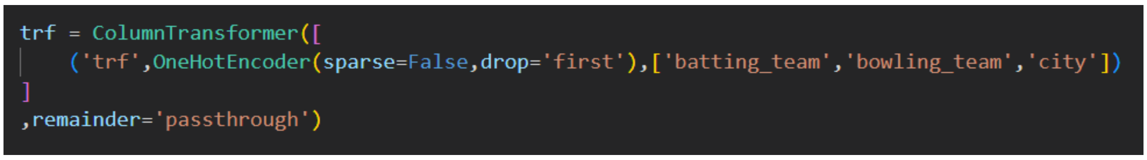
Activity 4: Importing required packages and encoding.

Import required packages.



```
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
import xgboost
from xgboost import XGBRegressor
from sklearn.metrics import r2_score,mean_absolute_error
```

Encode the data using one hot encoding and column transfer method.

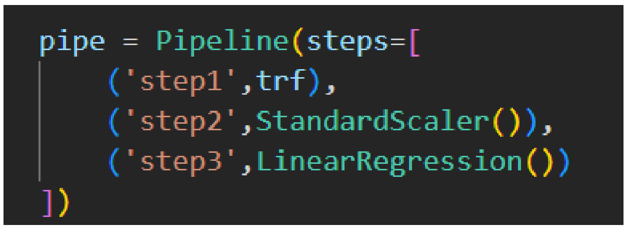


```
trf = ColumnTransformer([
    ('trf',OneHotEncoder(sparse=False,drop='first'),['batting_team','bowling_team','city'])
]
,remainder='passthrough')
```

Milestone 4: Model Training

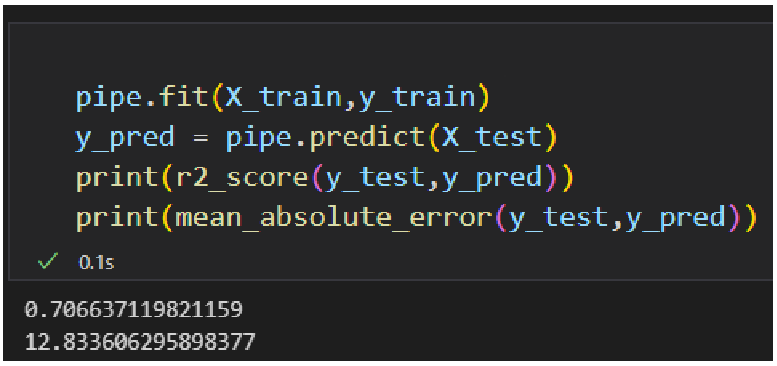
Model 1: Linear Regression

Creating pipeline



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

Training and calculating the accuracy.

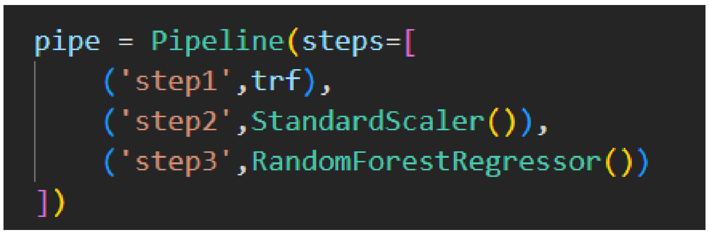


```
pipe.fit(X_train,y_train)
y_pred = pipe.predict(X_test)
print(r2_score(y_test,y_pred))
print(mean_absolute_error(y_test,y_pred))
✓ 0.1s

0.706637119821159
12.833606295898377
```

The accuracy is around 70%.

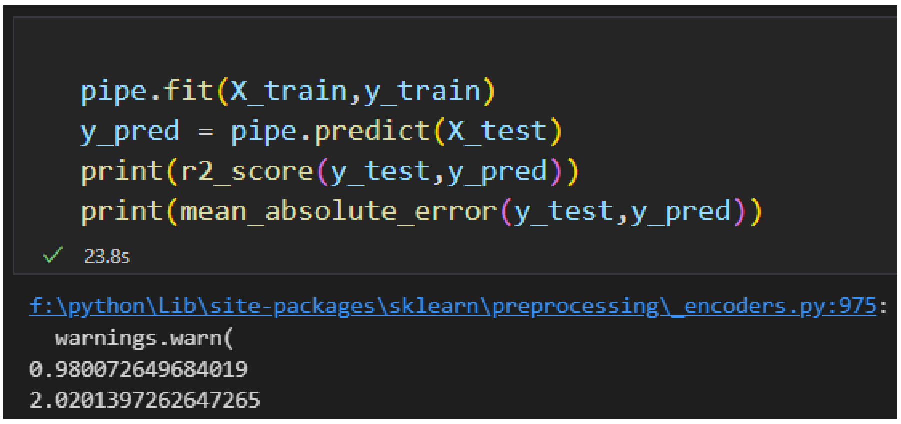
Model 2: Random Forest Regressor

Creating pipeline



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

Training and calculating the accuracy.

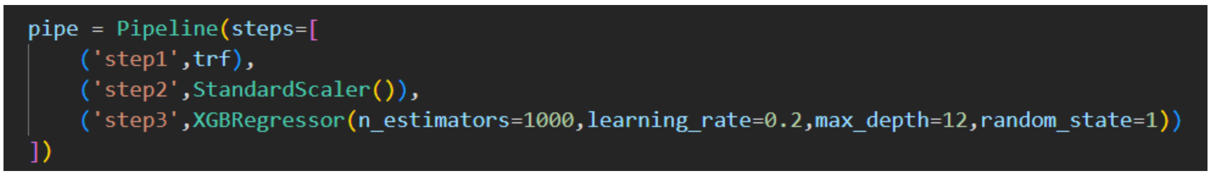


```
pipe.fit(X_train,y_train)
y_pred = pipe.predict(X_test)
print(r2_score(y_test,y_pred))
print(mean_absolute_error(y_test,y_pred))
✓ 23.8s

f:\python\Lib\site-packages\sklearn\preprocessing\_encoders.py:975:
  warnings.warn(
0.980072649684019
2.0201397262647265
```

The accuracy is around 98%.

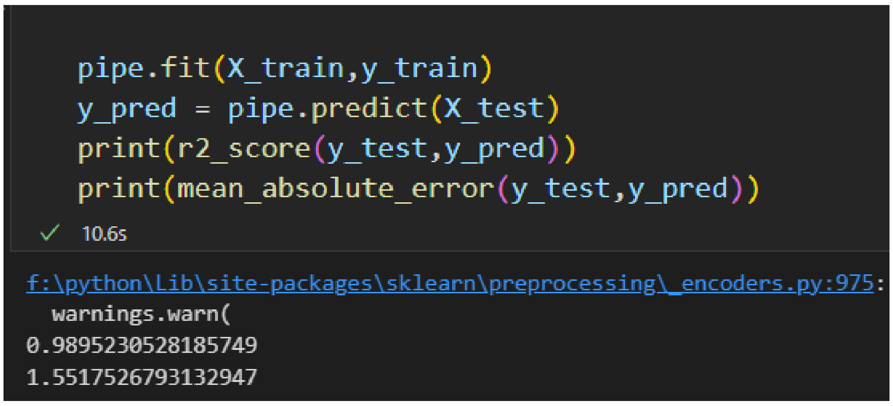
Model 3: XGBRegressor

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))
```

```
✓ 10.6s

f:\python\Lib\site-packages\sklearn\preprocessing\_encoders.py:975:
  warnings.warn(
0.9895230528185749
1.5517526793132947
```

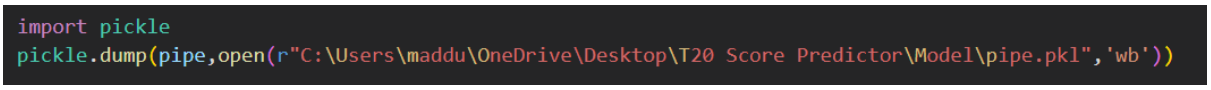
The accuracy is near to 99%.

Milestone 5: Model Deployment

Activity 1: Save the best model

After analysing 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.



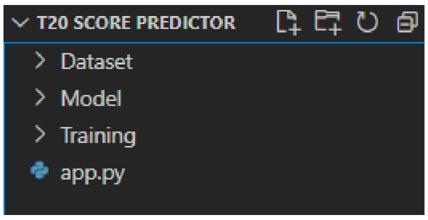
```
import pickle
pickle.dump(pipe,open(r"C:\Users\maddu\OneDrive\Desktop\T20 Score Predictor\Model\pipe.pkl",'wb'))
```

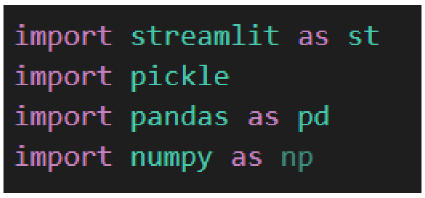
In this project, we will be building a web application that is integrated to the model we built.

We are 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.

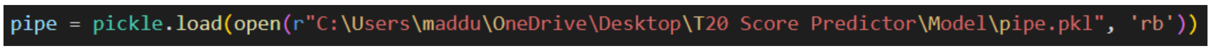
Activity 2: Create a app.py file and import necessary packages:



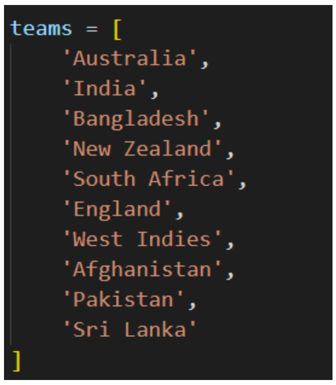


```python
import streamlit as st
import pickle
import pandas as pd
import numpy as np
```

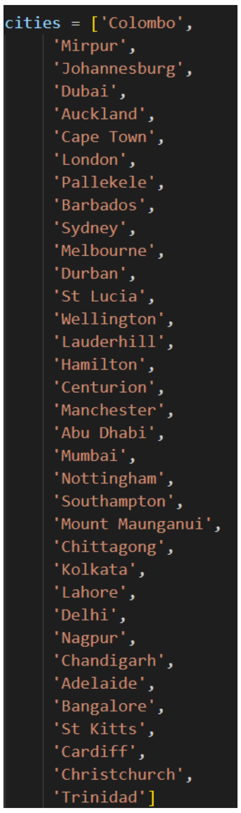
Activity 3: Defining teams/cities names and loading the pipeline:



```python
pipe = pickle.load(open(r"C:\Users\maddu\OneDrive\Desktop\T20 Score Predictor\Model\pipe.pkl", 'rb'))
```

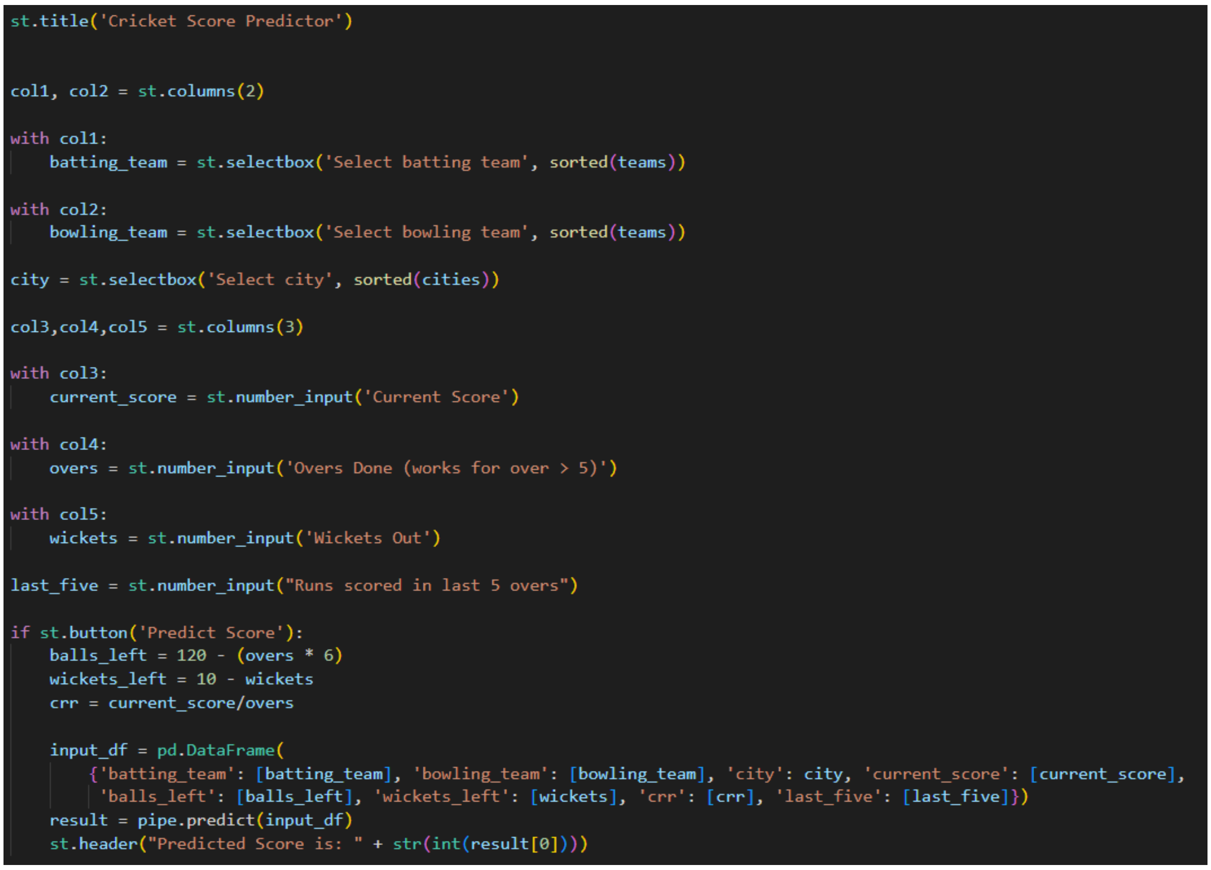


```python
teams = [
    'Australia',
    'India',
    'Bangladesh',
    'New Zealand',
    'South Africa',
    'England',
    'West Indies',
    'Afghanistan',
    'Pakistan',
    'Sri Lanka'
]
```



```python
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']
```

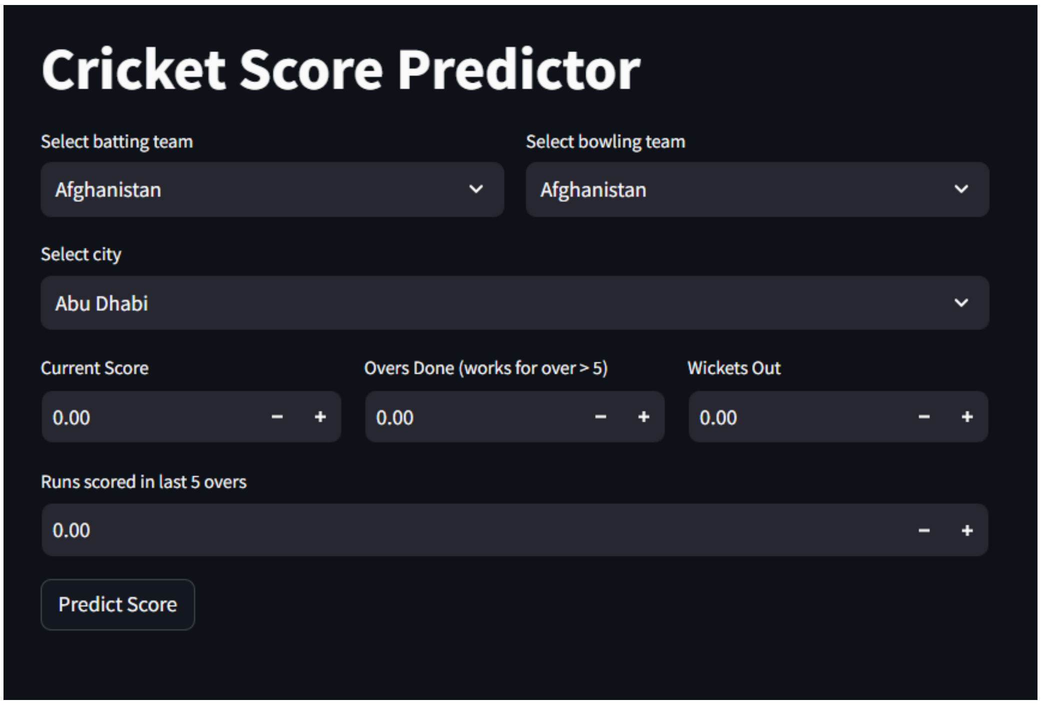
Activity 4: Accepting the input from user and prediction



```python
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 is: " + str(int(result[0])))
```

Output:

Website before entering the data:



Cricket Score Predictor

Select batting team: Afghanistan

Select bowling team: Afghanistan

Select city: Abu Dhabi

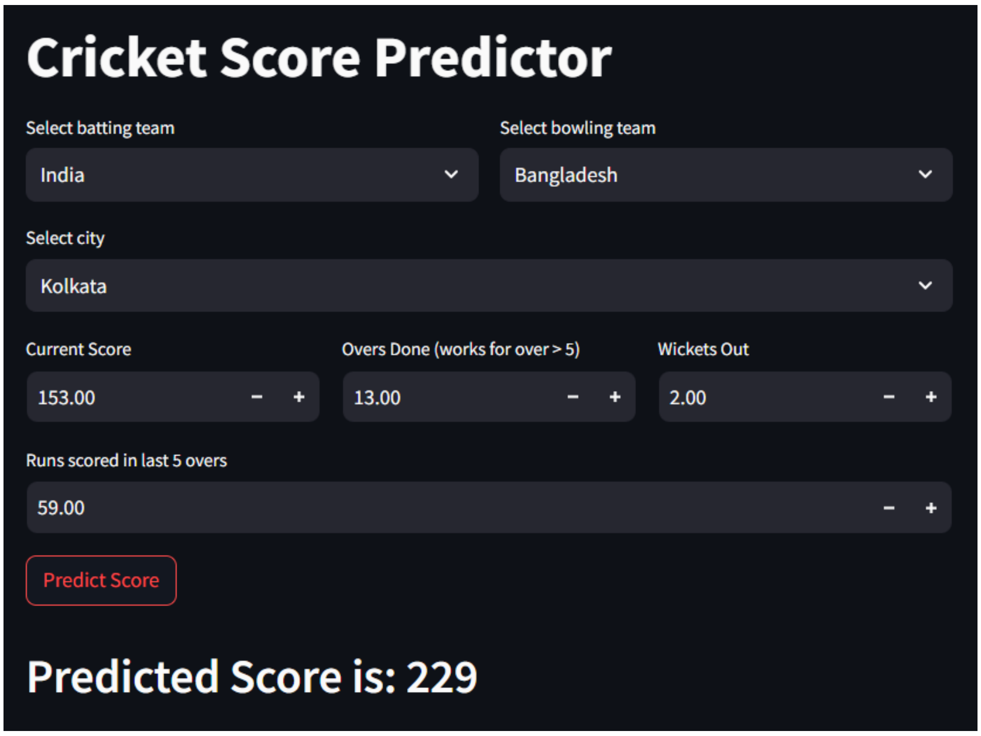
Current Score: 0.00

Overs Done (works for over > 5): 0.00

Wickets Out: 0.00

Runs scored in last 5 overs: 0.00

After entering the data:



Cricket Score Predictor

Select batting team: India

Select bowling team: Bangladesh

Select city: Kolkata

Current Score: 153.00

Overs Done (works for over > 5): 13.00

Wickets Out: 2.00

Runs scored in last 5 overs: 59.00

Predicted Score is: 229