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.

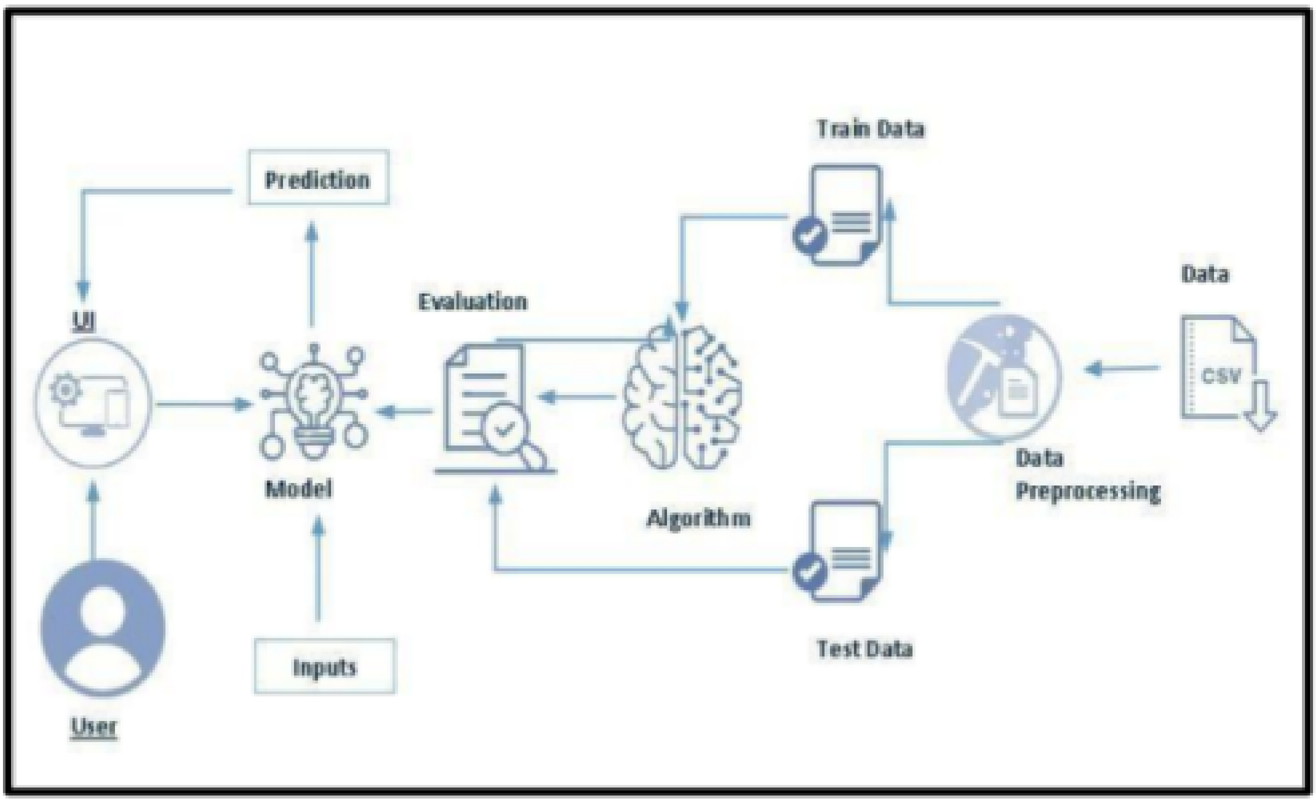
2. **Strategic Decision Making:** Cricket teams, coaches, and analysts can use the predictive model to make informed decisions during matches. This could involve adjusting strategies based on predicted scores, understanding the impact of different factors on the outcome, and optimizing team compositions.

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:



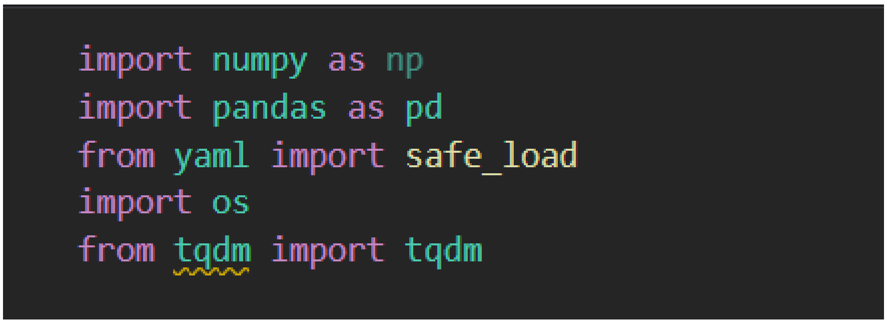
Data Collection & Preparation

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.

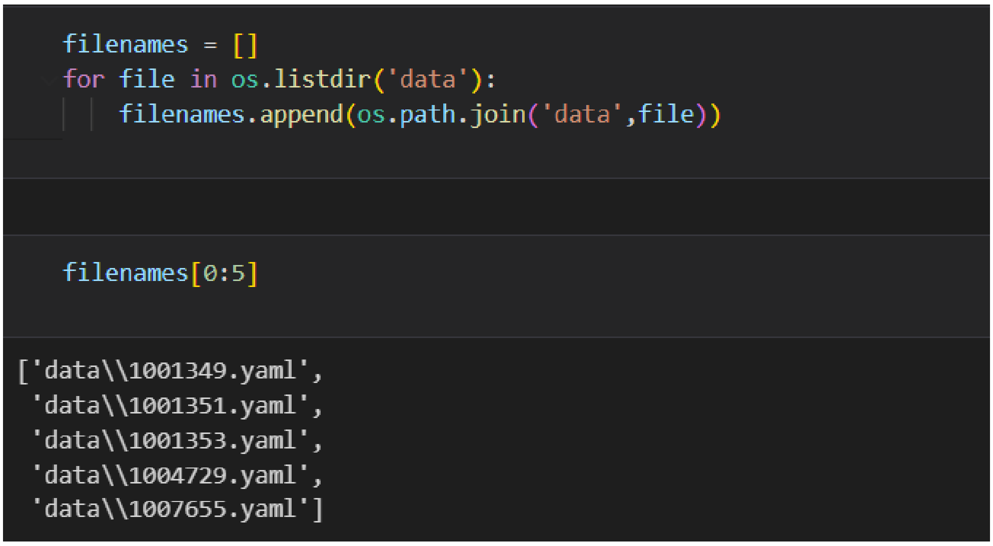
Importing the libraries

Import the necessary libraries as shown in the image.



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

Converting .yaml files to dataframes.

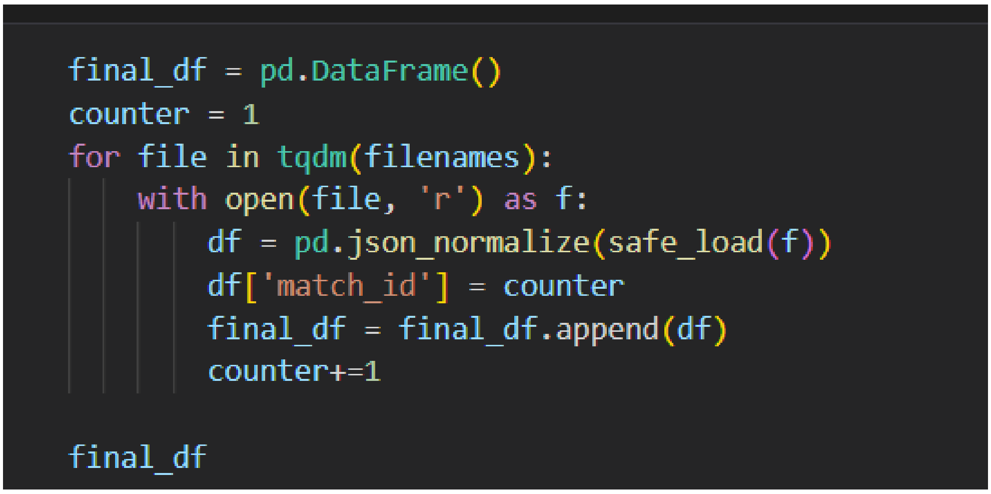


```
filenames = []
for file in os.listdir('data'):
    filenames.append(os.path.join('data',file))
```

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

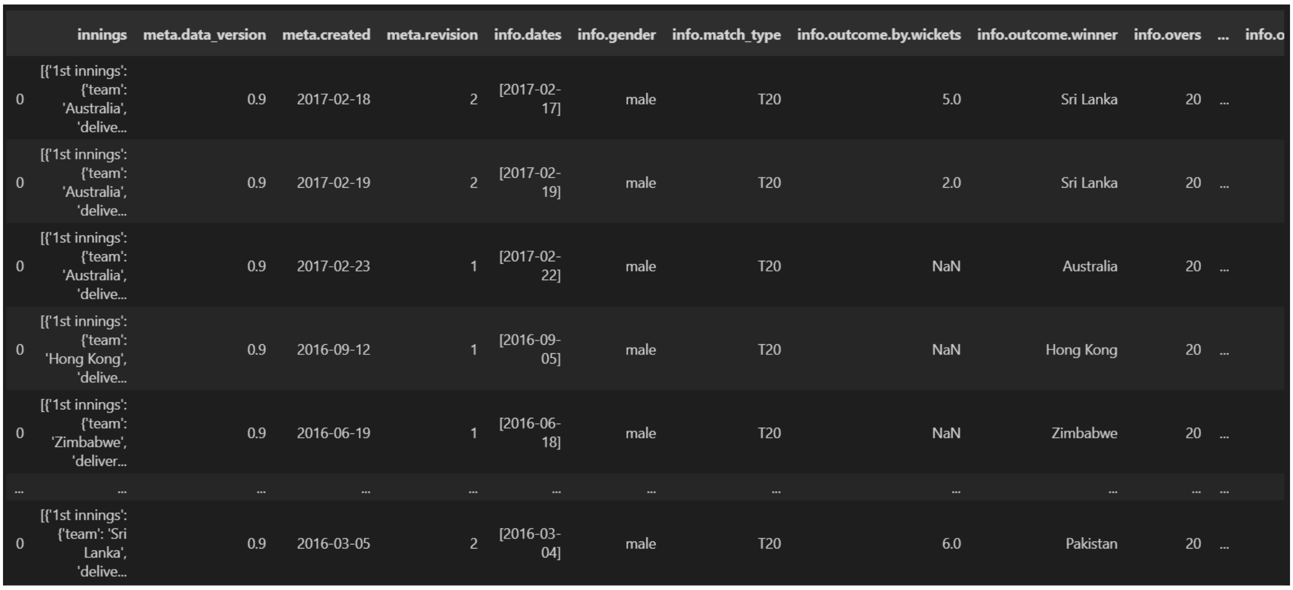
```
['data\\1001349.yaml',
 'data\\1001351.yaml',
 'data\\1001353.yaml',
 'data\\1004729.yaml',
 'data\\1007655.yaml']
```

Conversion :



```
final_df = pd.DataFrame()
counter = 1
for file in tqdm(filenames):
    with open(file, 'r') as f:
        df = pd.json_normalize(safe_load(f))
        df['match_id'] = counter
        final_df = final_df.append(df)
        counter+=1

final_df
```

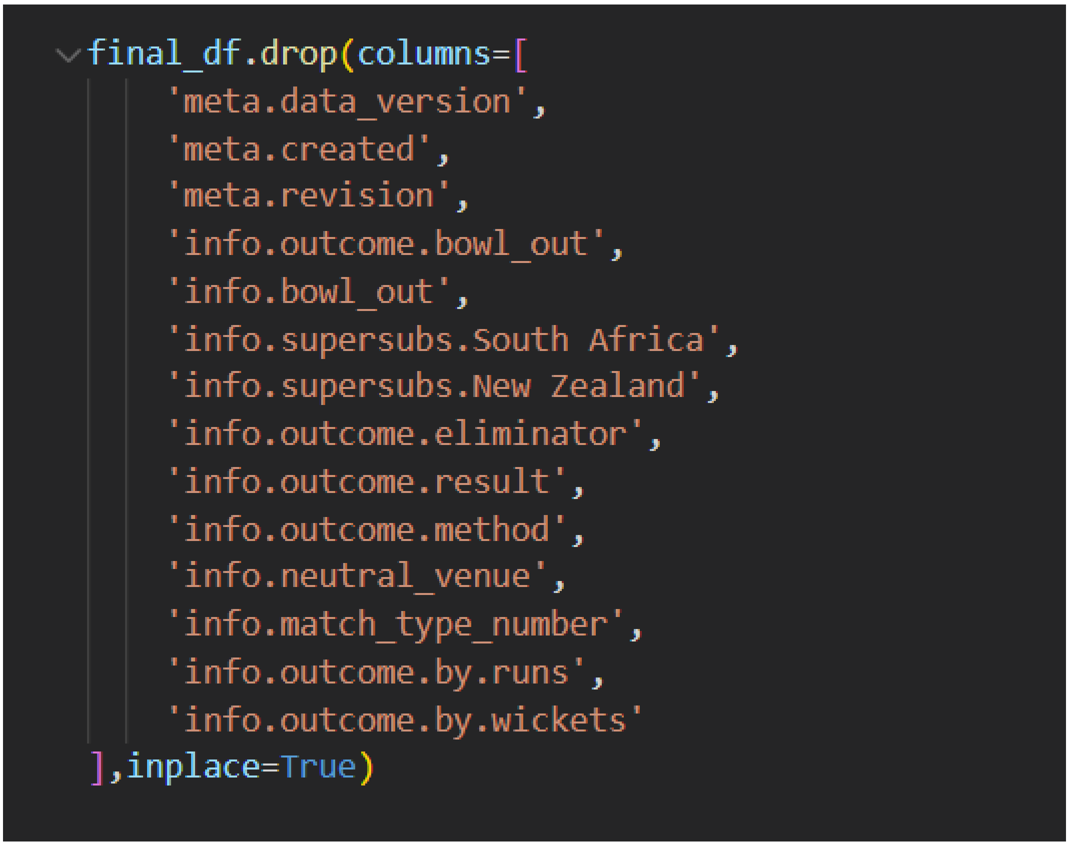
Below shown dataframe should be you output.



	innings	meta.data_version	meta.created	meta.revision	info.dates	info.gender	info.match_type	info.outcome.by.wickets	info.outcome.winner	info.overs	...	info.o
0	[{'1st innings': {'team': 'Australia', 'delive...	0.9	2017-02-18	2	[2017-02-17]	male	T20	5.0	Sri Lanka	20	...	
0	[{'1st innings': {'team': 'Australia', 'delive...	0.9	2017-02-19	2	[2017-02-19]	male	T20	2.0	Sri Lanka	20	...	
0	[{'1st innings': {'team': 'Australia', 'delive...	0.9	2017-02-23	1	[2017-02-22]	male	T20	NaN	Australia	20	...	
0	[{'1st innings': {'team': 'Hong Kong', 'delive...	0.9	2016-09-12	1	[2016-09-05]	male	T20	NaN	Hong Kong	20	...	
0	[{'1st innings': {'team': 'Zimbabwe', 'deliver...	0.9	2016-06-19	1	[2016-06-18]	male	T20	NaN	Zimbabwe	20	...	
...	...	...	...	...	...	...	...	...	...	...	...	
0	[{'1st innings': {'team': 'Sri Lanka', 'delive...	0.9	2016-03-05	2	[2016-03-04]	male	T20	6.0	Pakistan	20	...	

Exploratory Data Analysis

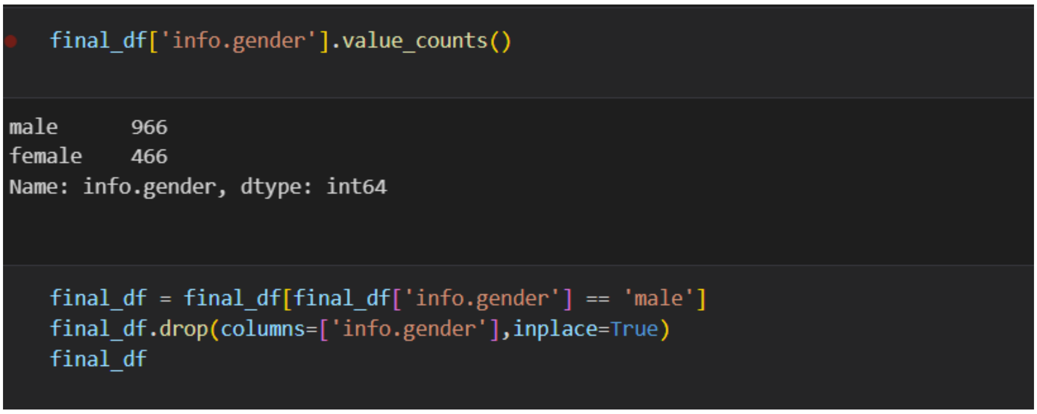
Activity 1: Dropping unnecessary columns



```
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.



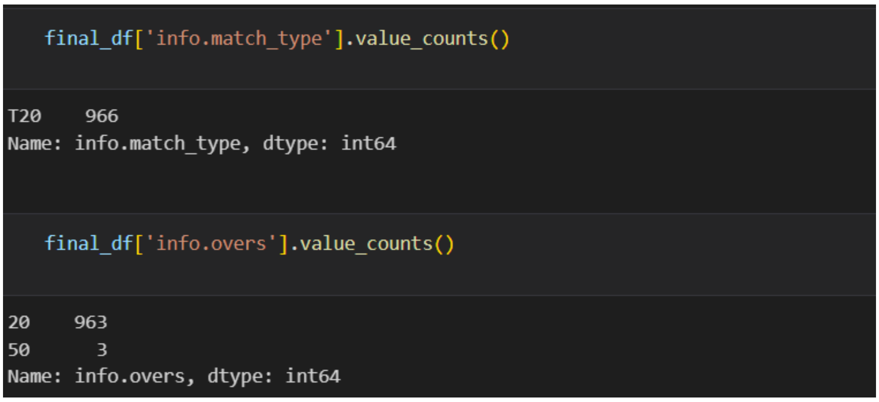
```
final_df['info.gender'].value_counts()
```

```
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.



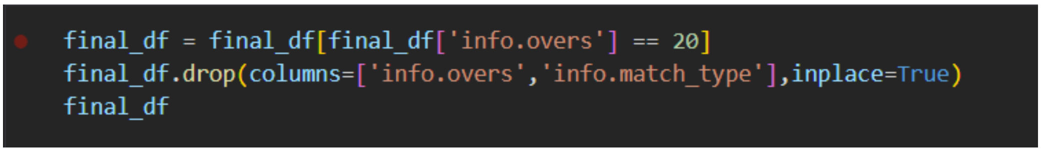
```
final_df['info.match_type'].value_counts()
```

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

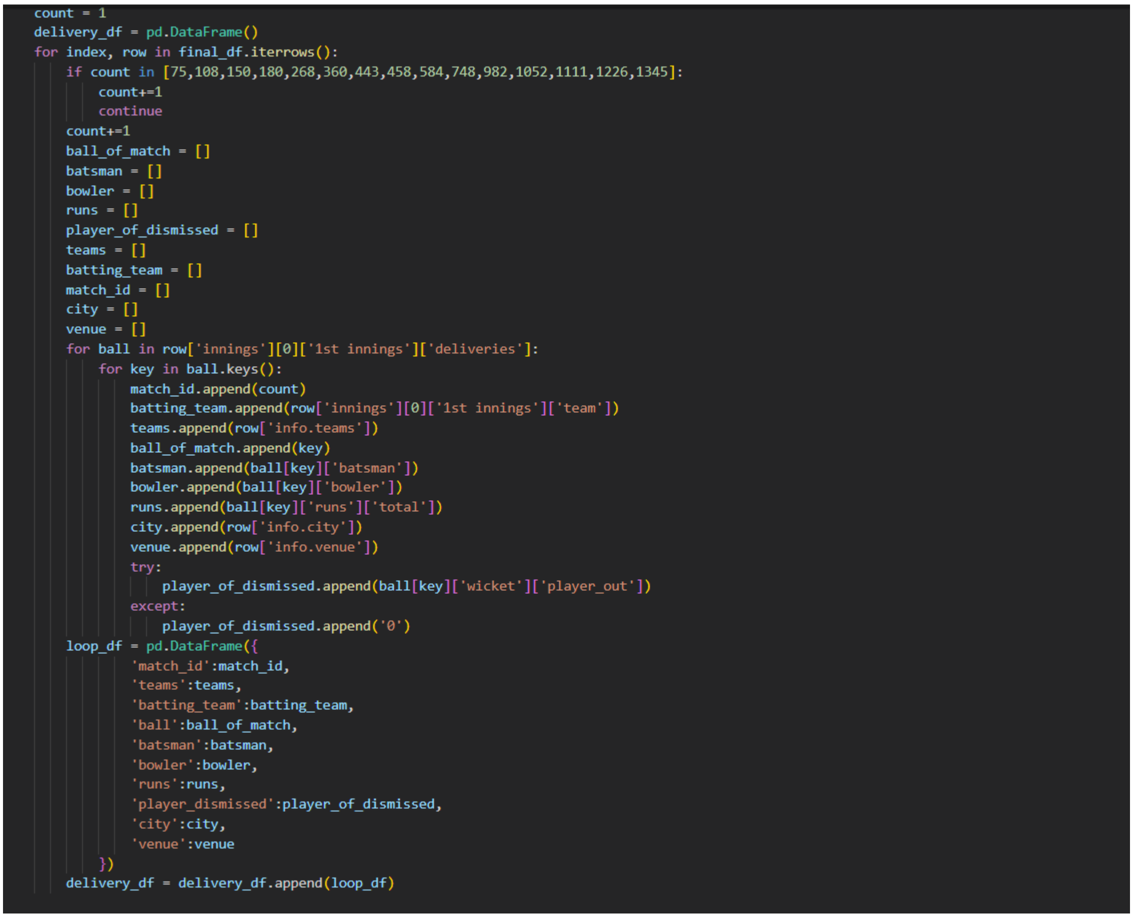
```
final_df['info.overs'].value_counts()
```

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

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



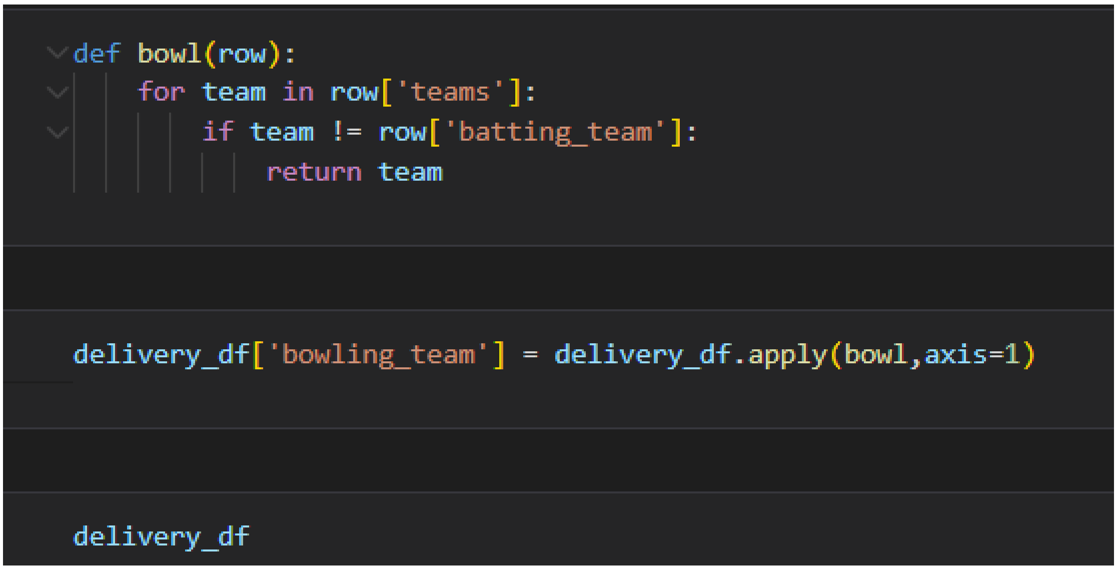
```
final_df = final_df[final_df['info.overs'] == 20]
final_df.drop(columns=['info.overs','info.match_type'],inplace=True)
final_df
```



```
count = 1
delivery_df = pd.DataFrame()
for index, row in final_df.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.

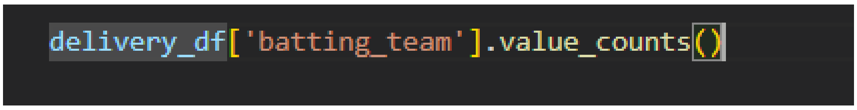


```
def bowl(row):
    for team in row['teams']:
        if team != row['batting_team']:
            return team
```

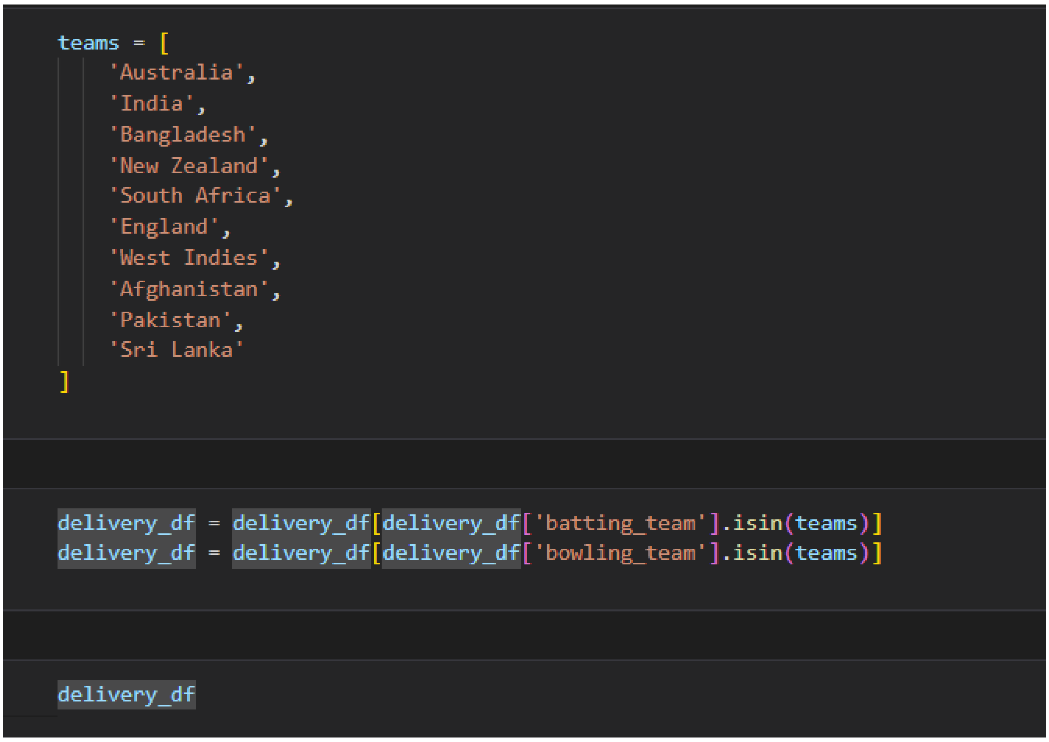
```
delivery_df['bowling_team'] = delivery_df.apply(bowl,axis=1)
```

```
delivery_df
```

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



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



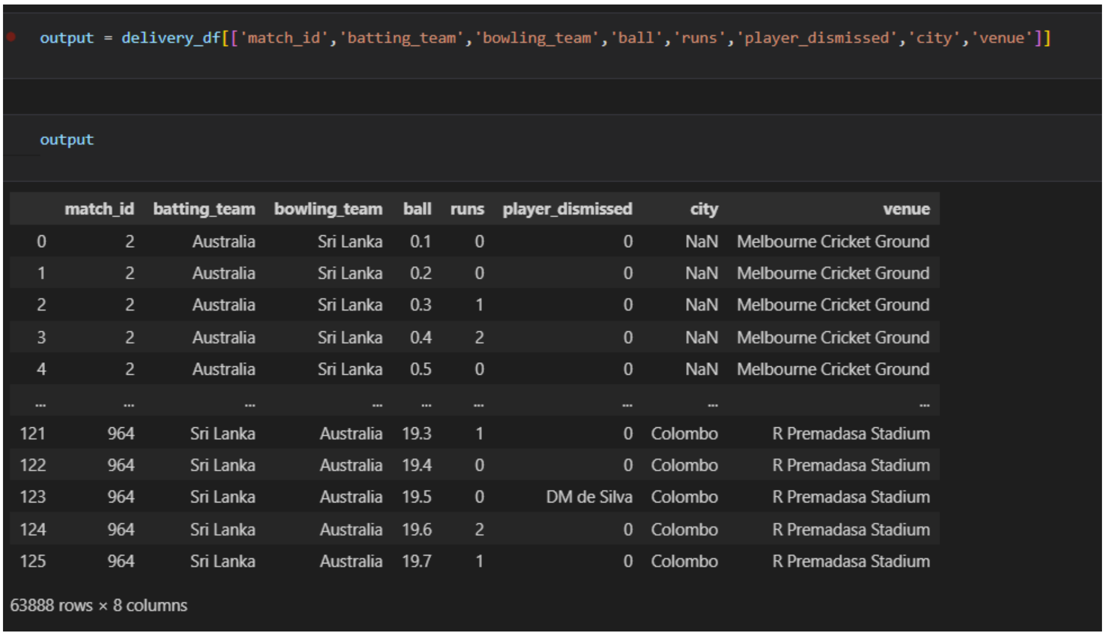
```
teams = [
    'Australia',
    'India',
    'Bangladesh',
    'New Zealand',
    'South Africa',
    'England',
    'West Indies',
    'Afghanistan',
    'Pakistan',
    'Sri Lanka'
]
```

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

```
delivery_df
```

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']]
```

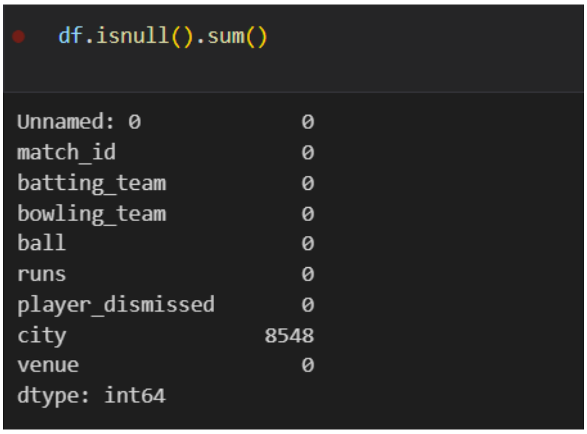
```
output
```

	match_id	batting_team	bowling_team	ball	runs	player_dismissed	city	venue
0	2	Australia	Sri Lanka	0.1	0	0	NaN	Melbourne Cricket Ground
1	2	Australia	Sri Lanka	0.2	0	0	NaN	Melbourne Cricket Ground
2	2	Australia	Sri Lanka	0.3	1	0	NaN	Melbourne Cricket Ground
3	2	Australia	Sri Lanka	0.4	2	0	NaN	Melbourne Cricket Ground
4	2	Australia	Sri Lanka	0.5	0	0	NaN	Melbourne Cricket Ground
...	...	...	...	...	...	...	...	...
121	964	Sri Lanka	Australia	19.3	1	0	Colombo	R Premadasa Stadium
122	964	Sri Lanka	Australia	19.4	0	0	Colombo	R Premadasa Stadium
123	964	Sri Lanka	Australia	19.5	0	DM de Silva	Colombo	R Premadasa Stadium
124	964	Sri Lanka	Australia	19.6	2	0	Colombo	R Premadasa Stadium
125	964	Sri Lanka	Australia	19.7	1	0	Colombo	R Premadasa Stadium

63888 rows × 8 columns

Activity 2: Feature Extraction

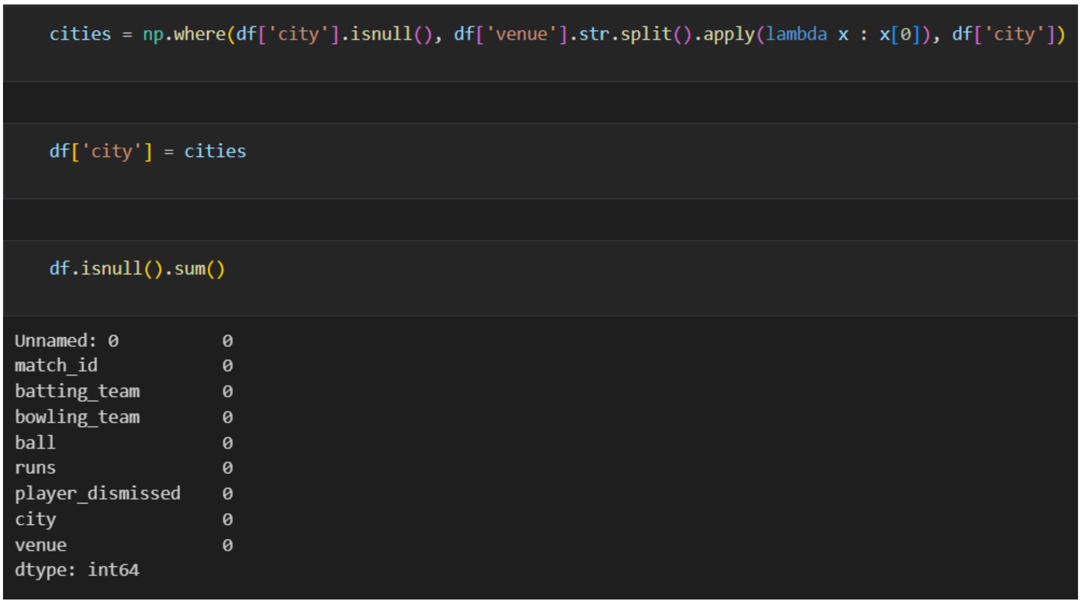
- Looking for null values



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

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

- Extracting city names using venue column and dropping venue column



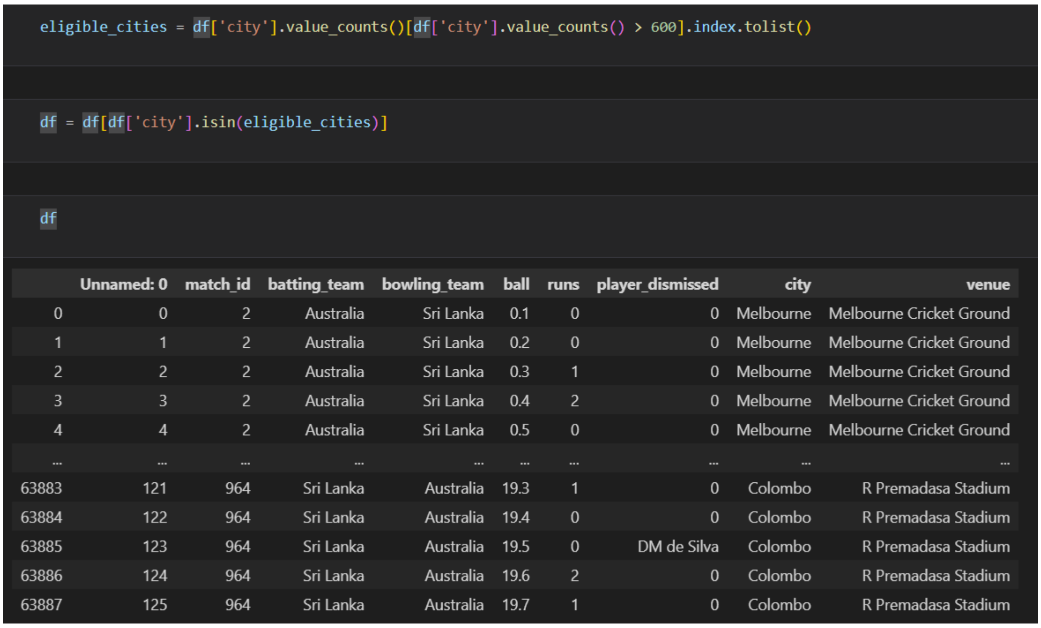
```
cities = np.where(df['city'].isnull(), df['venue'].str.split().apply(lambda x : x[0]), df['city'])
```

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

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

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

- 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.



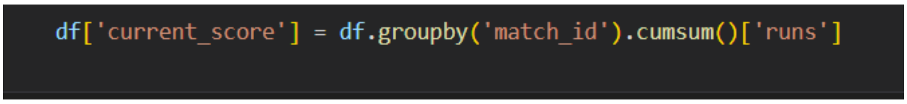
```
eligible_cities = df['city'].value_counts()[df['city'].value_counts() > 600].index.tolist()
```

```
df = df[df['city'].isin(eligible_cities)]
```

```
df
```

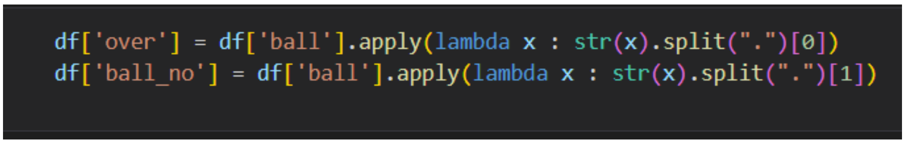
	Unnamed: 0	match_id	batting_team	bowling_team	ball	runs	player_dismissed	city	venue
0	0	2	Australia	Sri Lanka	0.1	0	0	Melbourne	Melbourne Cricket Ground
1	1	2	Australia	Sri Lanka	0.2	0	0	Melbourne	Melbourne Cricket Ground
2	2	2	Australia	Sri Lanka	0.3	1	0	Melbourne	Melbourne Cricket Ground
3	3	2	Australia	Sri Lanka	0.4	2	0	Melbourne	Melbourne Cricket Ground
4	4	2	Australia	Sri Lanka	0.5	0	0	Melbourne	Melbourne Cricket Ground
...	...	...	...	...	...	...	...	...	...
63883	121	964	Sri Lanka	Australia	19.3	1	0	Colombo	R Premadasa Stadium
63884	122	964	Sri Lanka	Australia	19.4	0	0	Colombo	R Premadasa Stadium
63885	123	964	Sri Lanka	Australia	19.5	0	DM de Silva	Colombo	R Premadasa Stadium
63886	124	964	Sri Lanka	Australia	19.6	2	0	Colombo	R Premadasa Stadium
63887	125	964	Sri Lanka	Australia	19.7	1	0	Colombo	R Premadasa Stadium

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



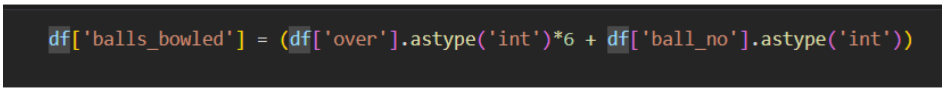
```
df['current_score'] = df.groupby('match_id').cumsum()['runs']
```

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

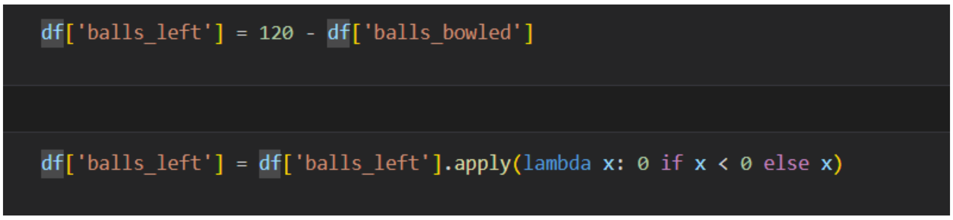


```
df['over'] = df['ball'].apply(lambda x : str(x).split(".")[0])
df['ball_no'] = df['ball'].apply(lambda x : str(x).split(".")[1])
```

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



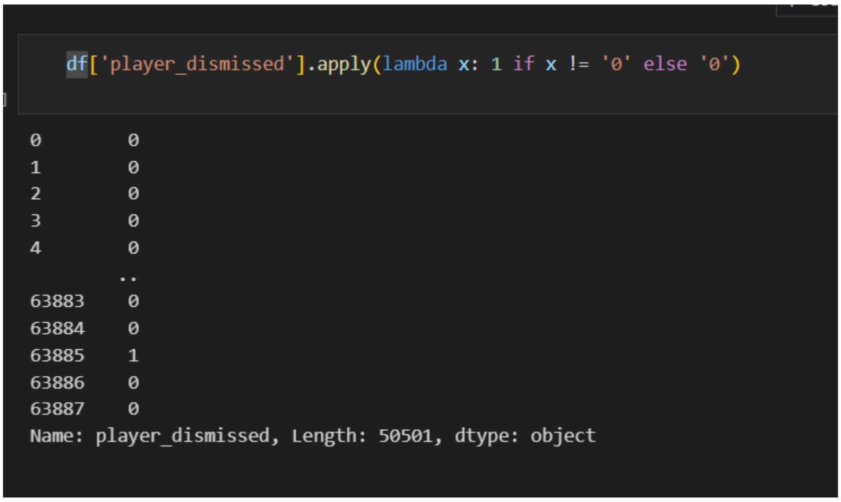
```
df['balls_bowled'] = (df['over'].astype('int')*6 + df['ball_no'].astype('int'))
```



```
df['balls_left'] = 120 - df['balls_bowled']
```

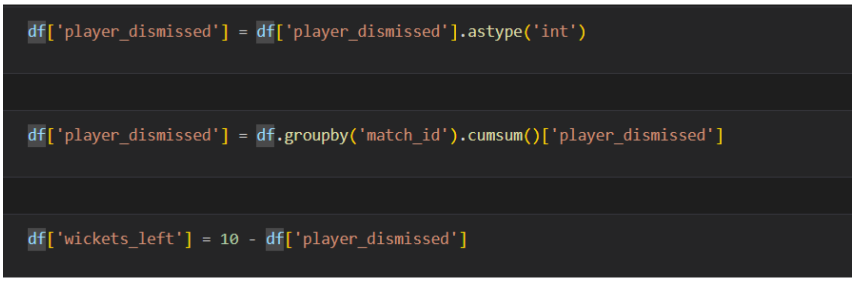
```
df['balls_left'] = df['balls_left'].apply(lambda x: 0 if x < 0 else x)
```

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



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

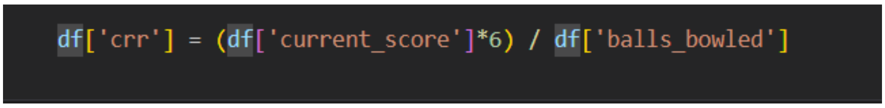
```
0        0
1        0
2        0
3        0
4        0
        ..
63883    0
63884    0
63885    1
63886    0
63887    0
Name: player_dismissed, Length: 50501, dtype: object
```



```
df['player_dismissed'] = df['player_dismissed'].astype('int')
```

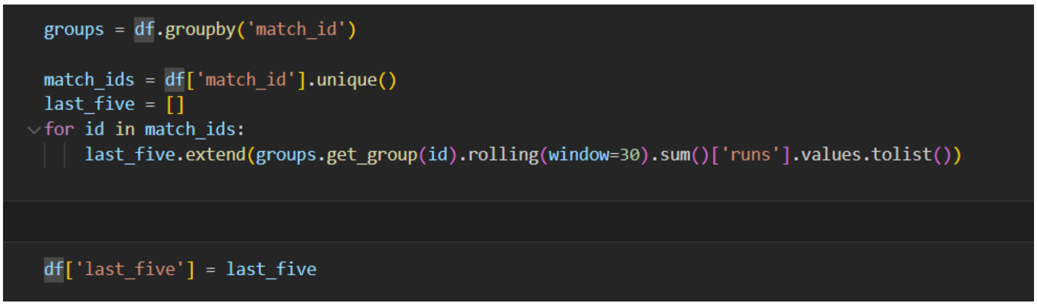
```
df['player_dismissed'] = df.groupby('match_id').cumsum()['player_dismissed']
```

```
df['wickets_left'] = 10 - df['player_dismissed']
```



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

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



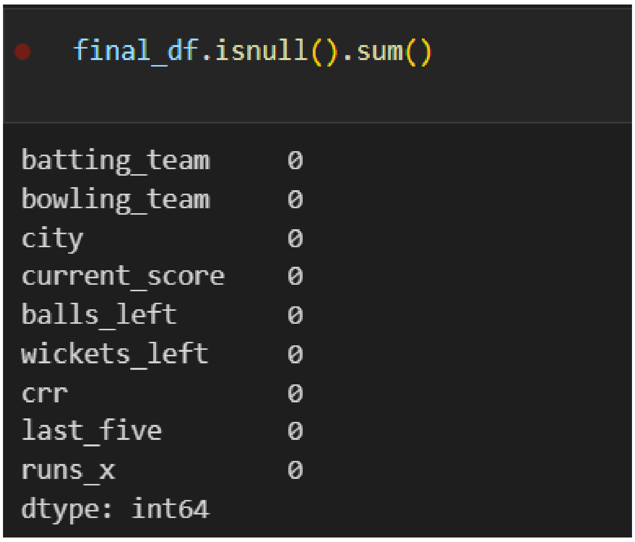
```
groups = df.groupby('match_id')

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
```

- 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']]

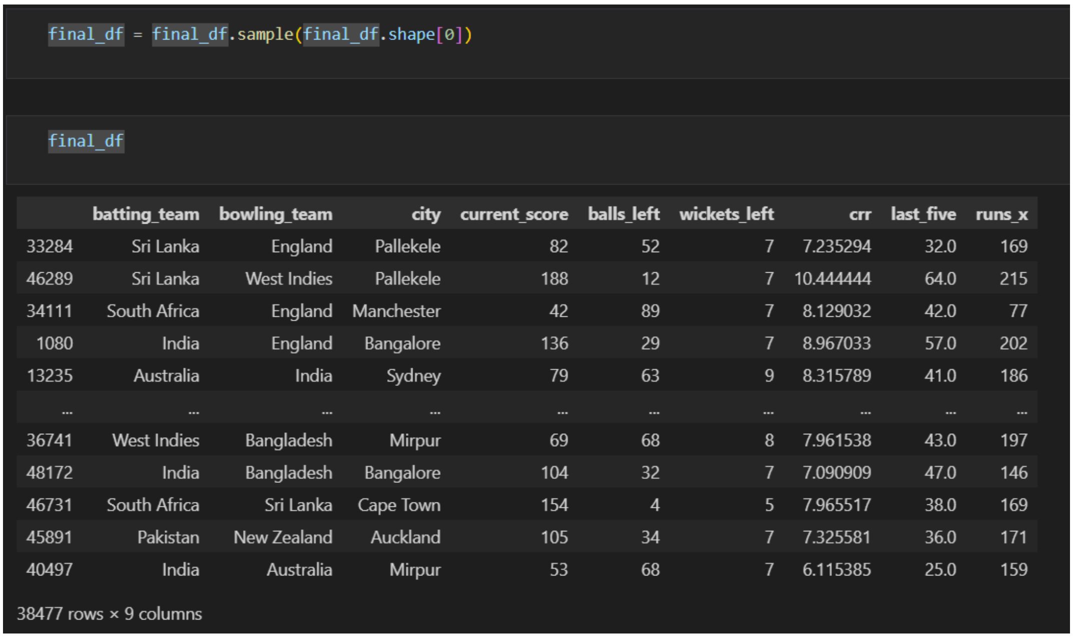
- Checking for null values in the final dataframe and drop them.



```
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        0
runs_x           0
dtype: int64
```

- Let's take a look at the final data frame which is ready for training.



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

```
final_df
```

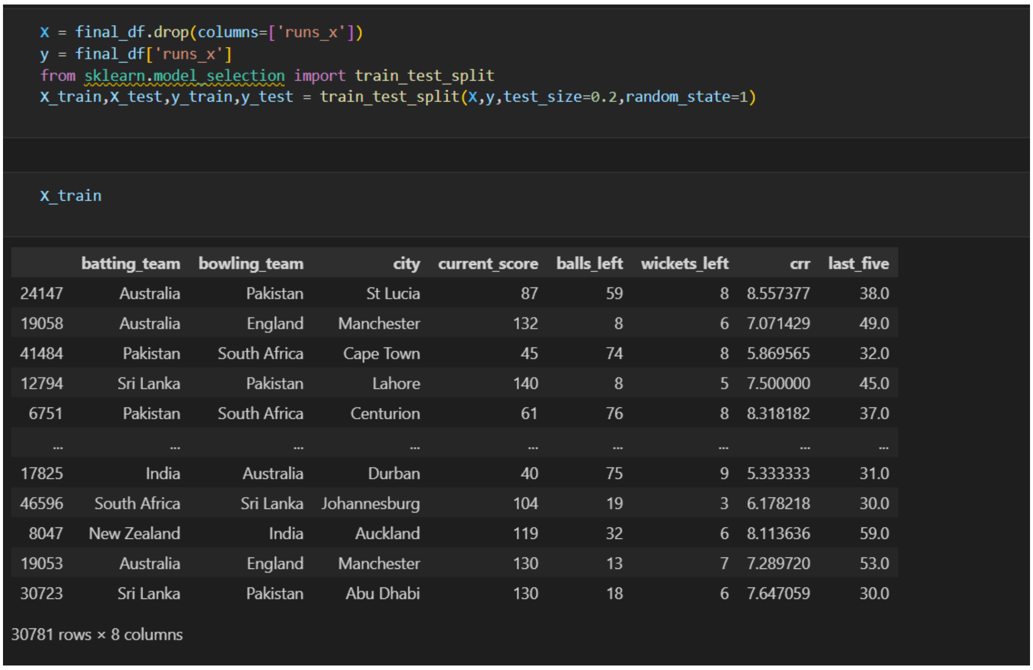
	batting_team	bowling_team	city	current_score	balls_left	wickets_left	crr	last_five	runs_x
33284	Sri Lanka	England	Pallekele	82	52	7	7.235294	32.0	169
46289	Sri Lanka	West Indies	Pallekele	188	12	7	10.444444	64.0	215
34111	South Africa	England	Manchester	42	89	7	8.129032	42.0	77
1080	India	England	Bangalore	136	29	7	8.967033	57.0	202
13235	Australia	India	Sydney	79	63	9	8.315789	41.0	186
...	...	...	...	...	...	...	...	...	...
36741	West Indies	Bangladesh	Mirpur	69	68	8	7.961538	43.0	197
48172	India	Bangladesh	Bangalore	104	32	7	7.090909	47.0	146
46731	South Africa	Sri Lanka	Cape Town	154	4	5	7.965517	38.0	169
45891	Pakistan	New Zealand	Auckland	105	34	7	7.325581	36.0	171
40497	India	Australia	Mirpur	53	68	7	6.115385	25.0	159

38477 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)
```

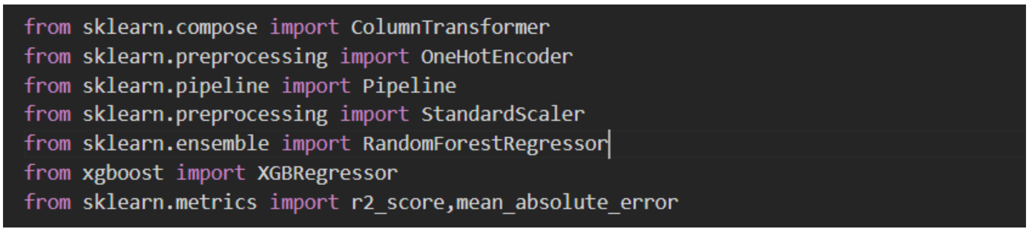
```
X_train
```

	batting_team	bowling_team	city	current_score	balls_left	wickets_left	crr	last_five
24147	Australia	Pakistan	St Lucia	87	59	8	8.557377	38.0
19058	Australia	England	Manchester	132	8	6	7.071429	49.0
41484	Pakistan	South Africa	Cape Town	45	74	8	5.869565	32.0
12794	Sri Lanka	Pakistan	Lahore	140	8	5	7.500000	45.0
6751	Pakistan	South Africa	Centurion	61	76	8	8.318182	37.0
...	...	...	...	...	...	...	...	...
17825	India	Australia	Durban	40	75	9	5.333333	31.0
46596	South Africa	Sri Lanka	Johannesburg	104	19	3	6.178218	30.0
8047	New Zealand	India	Auckland	119	32	6	8.113636	59.0
19053	Australia	England	Manchester	130	13	7	7.289720	53.0
30723	Sri Lanka	Pakistan	Abu Dhabi	130	18	6	7.647059	30.0

30781 rows × 8 columns

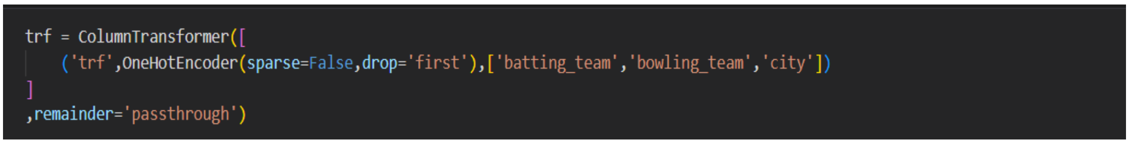
Activity 4: Importing required packages and Encoding.

- 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 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')
```

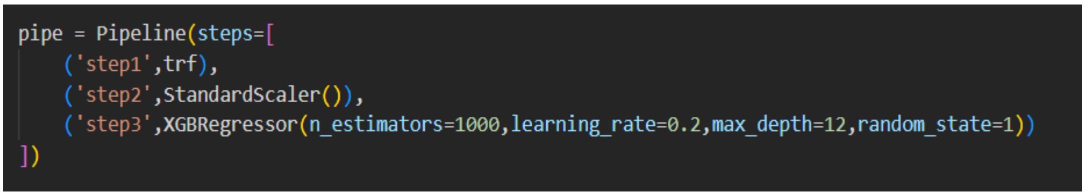
Milestone 4: Model Training

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.

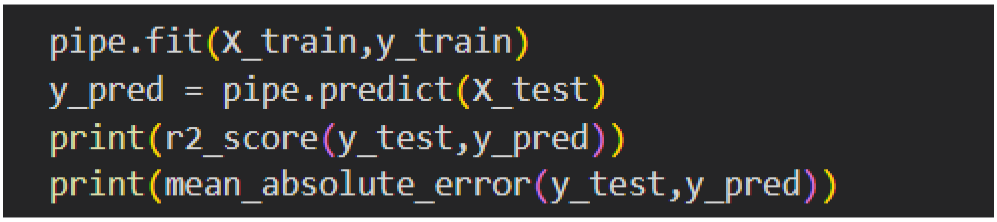
Model 1: Linear Regression

Creating pipeline

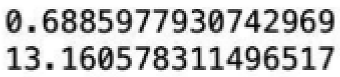


```
pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),
    ('step3',XGBRegressor(n_estimators=1000,learning_rate=0.2,max_depth=12,random_state=1))
])
```

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))
```

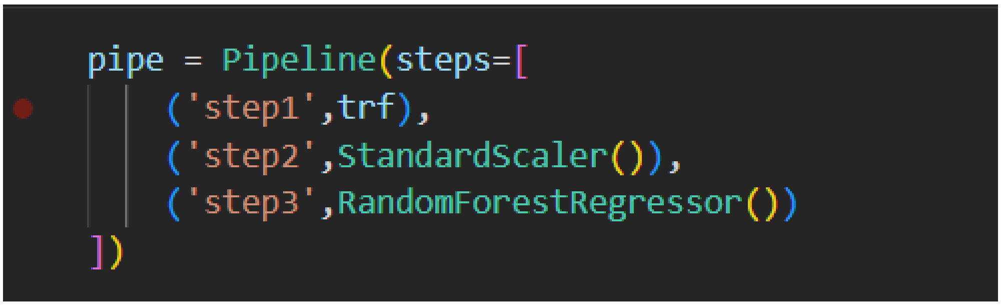


```
0.6885977930742969
13.160578311496517
```

Accuracy is near 68%

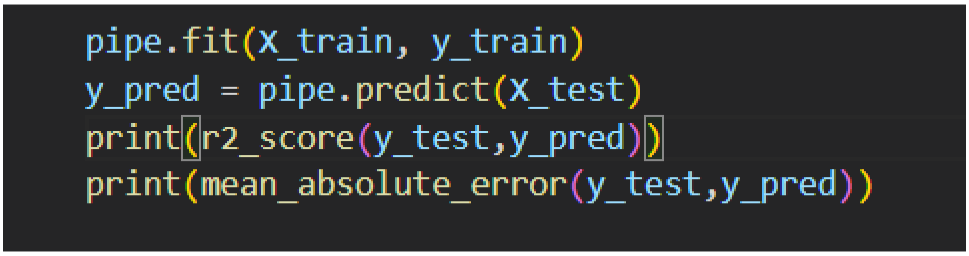
Model 2: Random Forest Regressor

Creating Pipeline

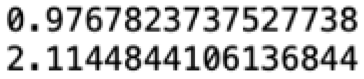


```
pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),
    ('step3',RandomForestRegressor())
])
```

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))
```

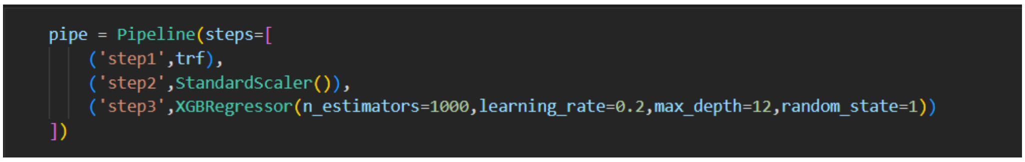


```
0.9767823737527738
2.1144844106136844
```

Accuracy is somewhere around 97%

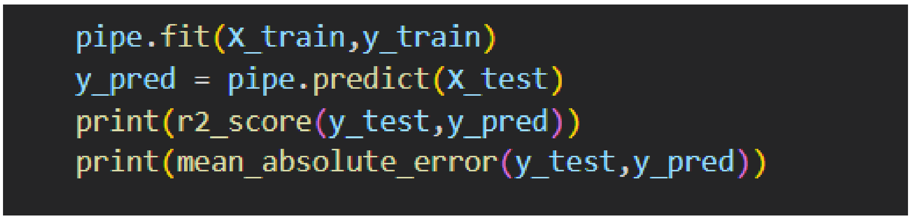
Model 3: XGBRegressor

Creating Pipeline

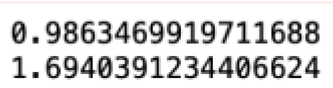


```
pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),
    ('step3',XGBRegressor(n_estimators=1000,learning_rate=0.2,max_depth=12,random_state=1))
])
```

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%.

Model Deployment

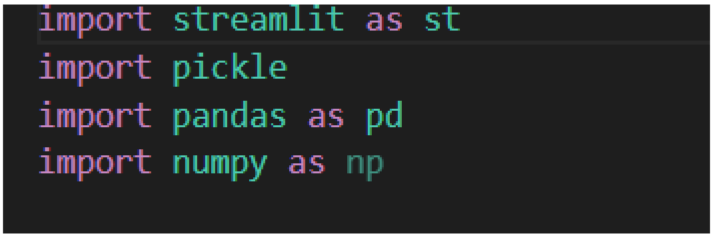
Integrate with Web Framework

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.

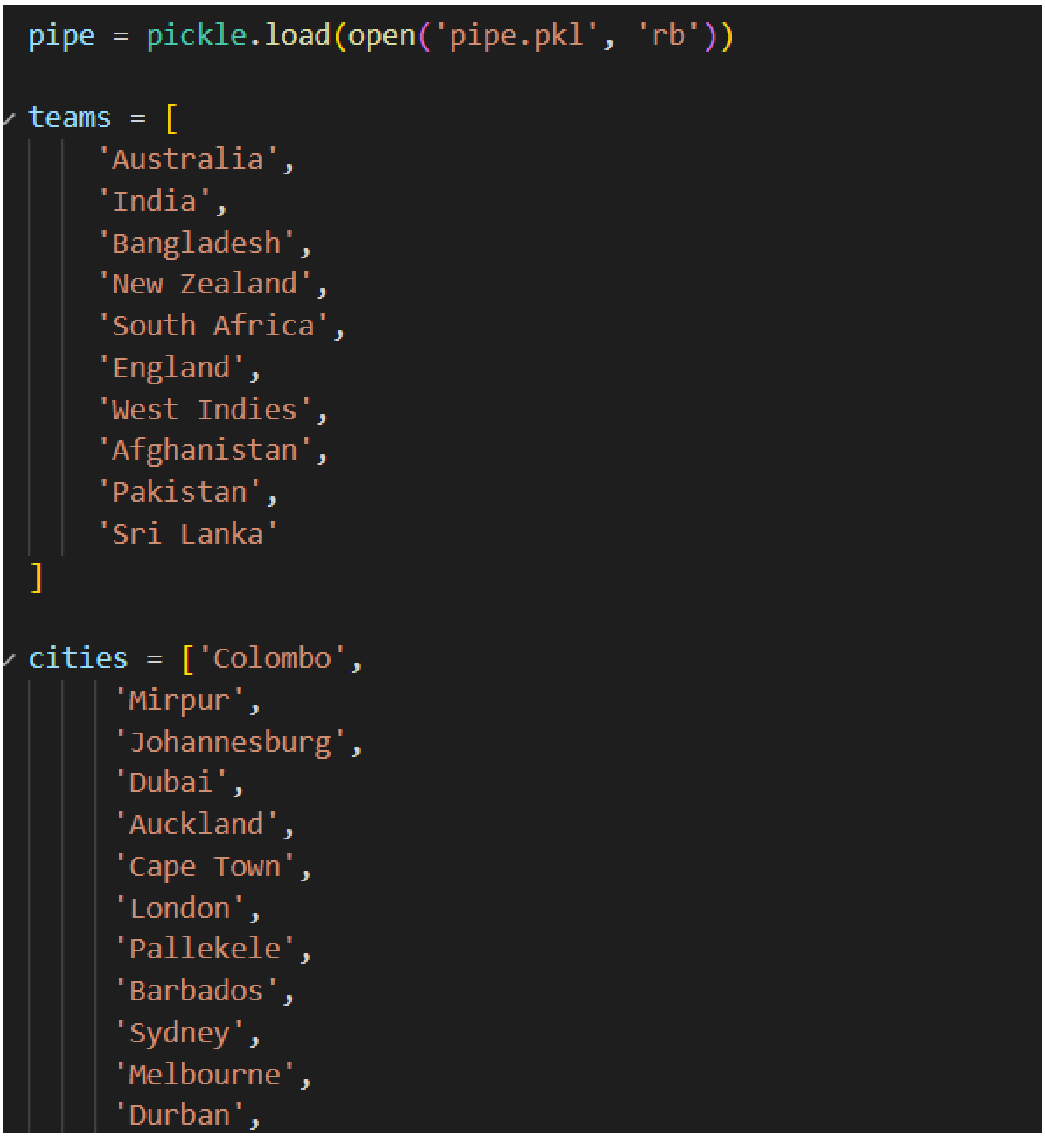
Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps.

Create a web.py file and import necessary packages:



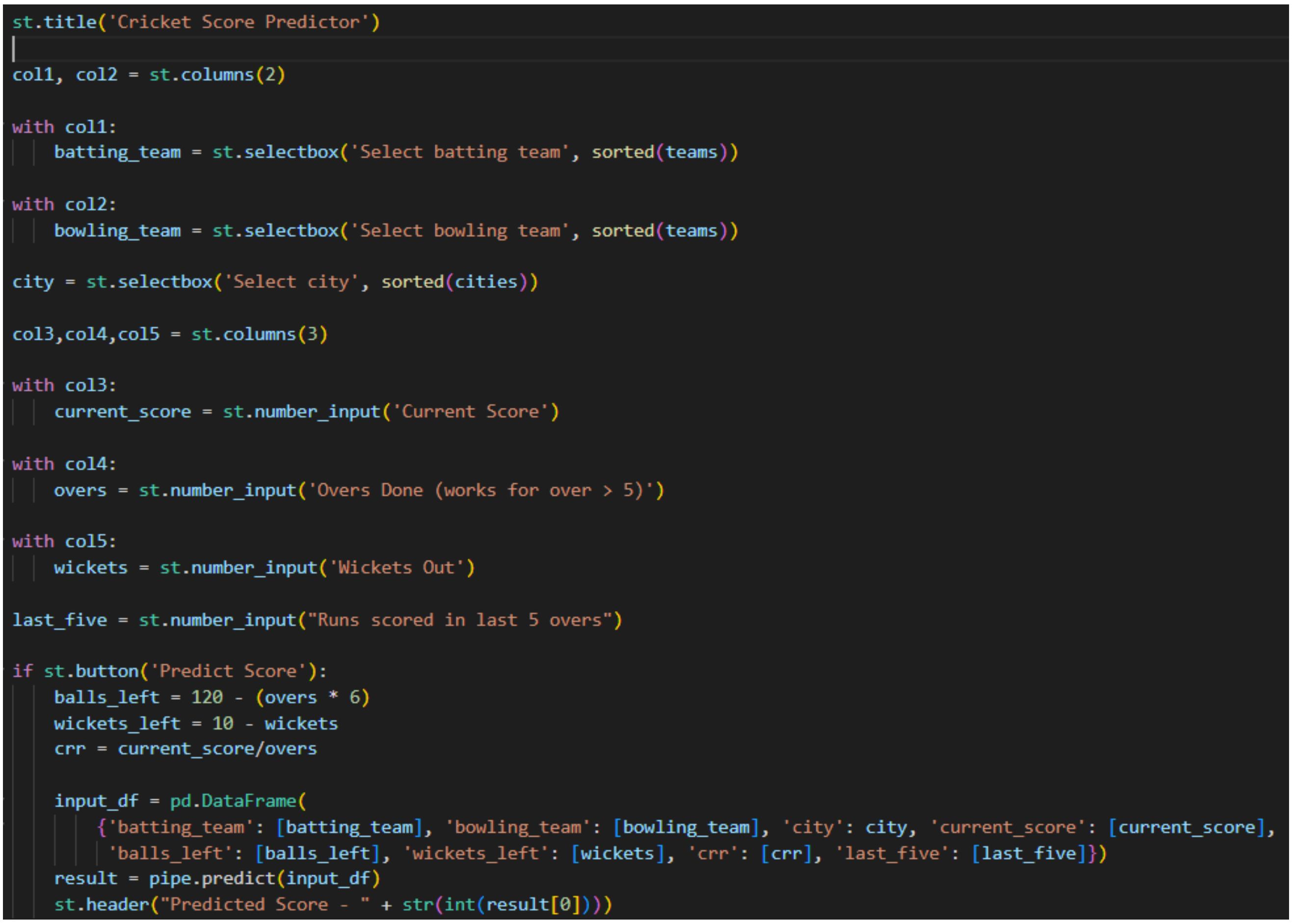
```
import streamlit as st
import pickle
import pandas as pd
import numpy as np
```

Activity 2.2: Defining teams/cities names and loading the pipeline:



```
pipe = pickle.load(open('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',
```

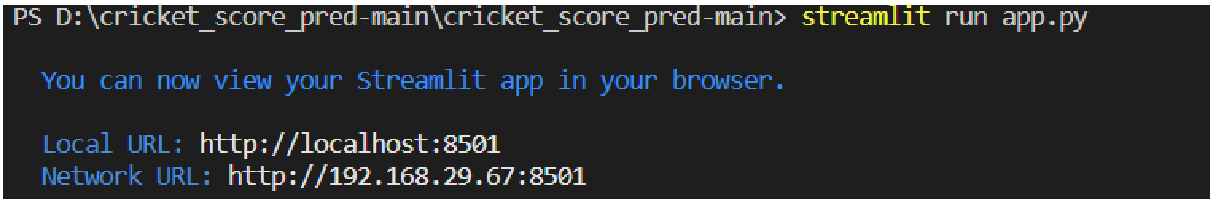
Activity 2.3: Accepting the input from user and prediction



```
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”

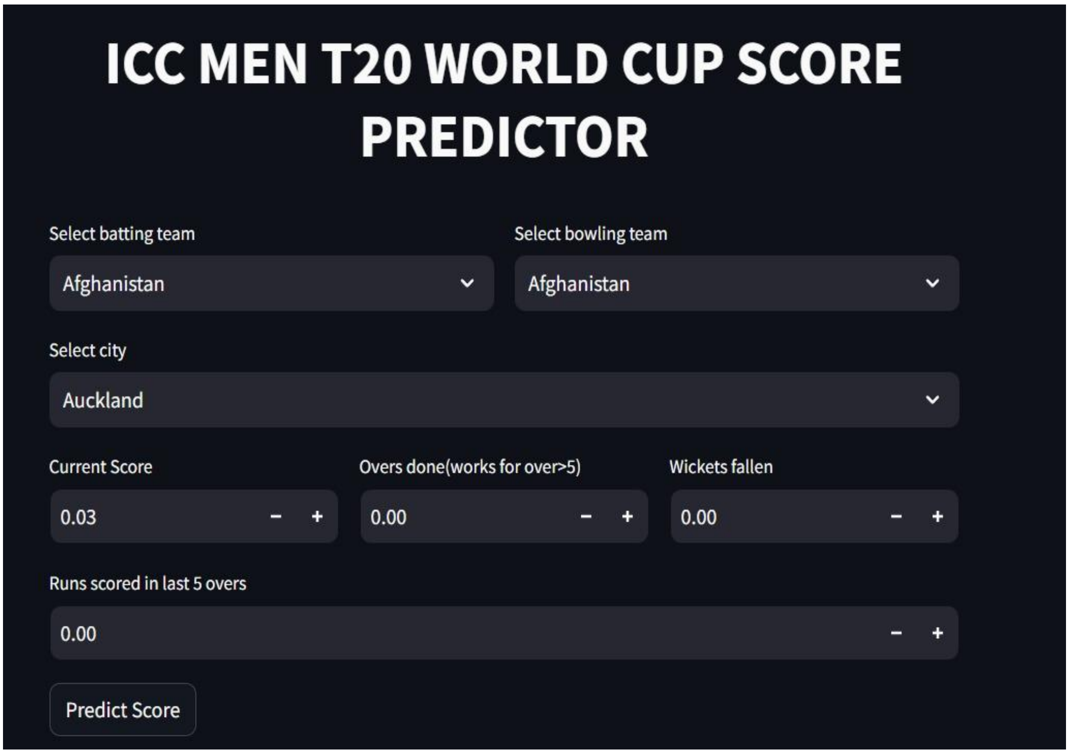


```
PS D:\cricket_score_pred-main\cricket_score_pred-main> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.29.67:8501
```

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

HOW DOES YOUR WEBSITE LOOK LIKE ?

- **Before entering the data :**



ICC MEN T20 WORLD CUP SCORE PREDICTOR

Select batting team: Afghanistan

Select bowling team: Afghanistan

Select city: Auckland

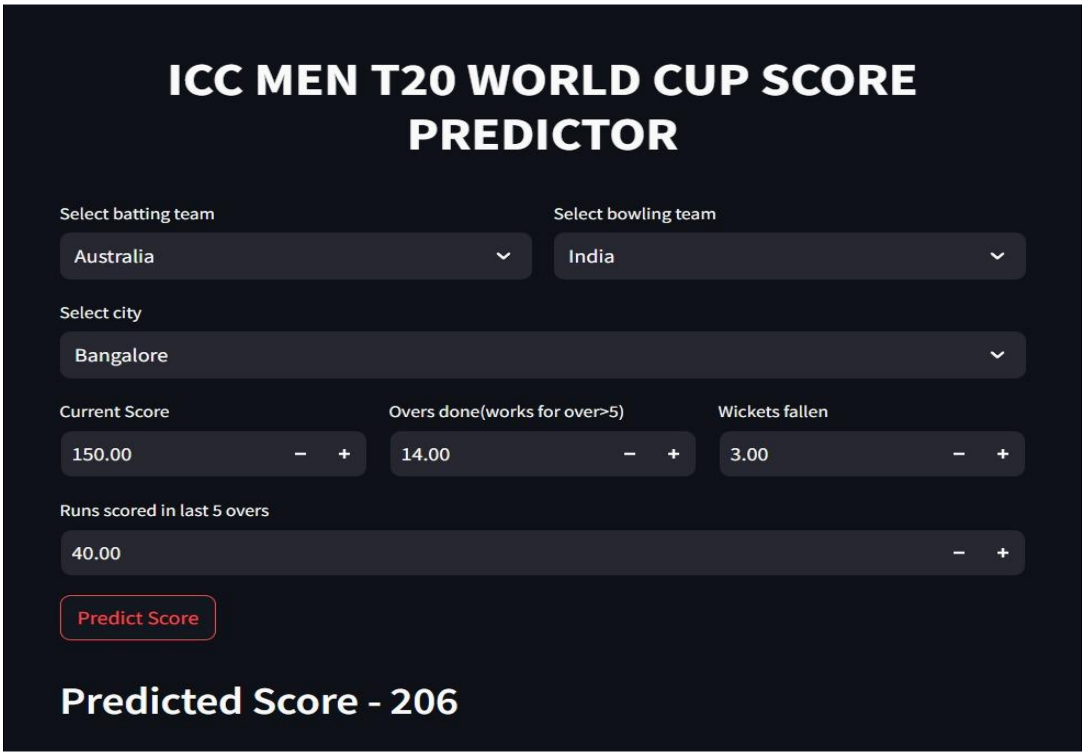
Current Score: 0.03

Overs done(works for over>5): 0.00

Wickets fallen: 0.00

Runs scored in last 5 overs: 0.00

- **After entering the data :**



ICC MEN T20 WORLD CUP SCORE PREDICTOR

Select batting team: Australia

Select bowling team: India

Select city: Bangalore

Current Score: 150.00

Overs done(works for over>5): 14.00

Wickets fallen: 3.00

Runs scored in last 5 overs: 40.00

Predicted Score - 206

<u>Project Demonstration & Documentation</u>

Record explanation Video for project end to end solution.

https://drive.google.com/file/d/1dAY1GqRDBZvKs6-D73J9Pcm6rOGxZ1EU/view?usp=sharing