HCLTech

Project report

Indian Premier League winning probability predictor system

Prepared by

Hamza

Jamia Hamdard university

Under supervision of

Ishita Pandey

software engineer at HCLTech

DECLARATION

I, **Mr. HAMZA** a student of **Bachelors in Technology Computer Science and Engineering with Artificial Intelligence (B.Tech C.S.E-AI)** hereby declare that the Project/Dissertation entitled **" Indian Premier League winning probability predictor system"** which is being submitted by me to the HCLTech, Noida in partial fulfillment of the requirement for the award of the internship certificate is my original work and has not been submitted anywhere else for the award of any Degree, Diploma, Associateship, Fellowship or other similar title or recognition.

HAMZA

Place: HCL headquarters sector 126 Noida.

ACKNOWLEDGEMENT

I would want to express my gratitude to a number of persons who have encouraged and assisted me in the preparation of this project, both directly and indirectly. It allows me to look back and reflect on the support I've had during this process.

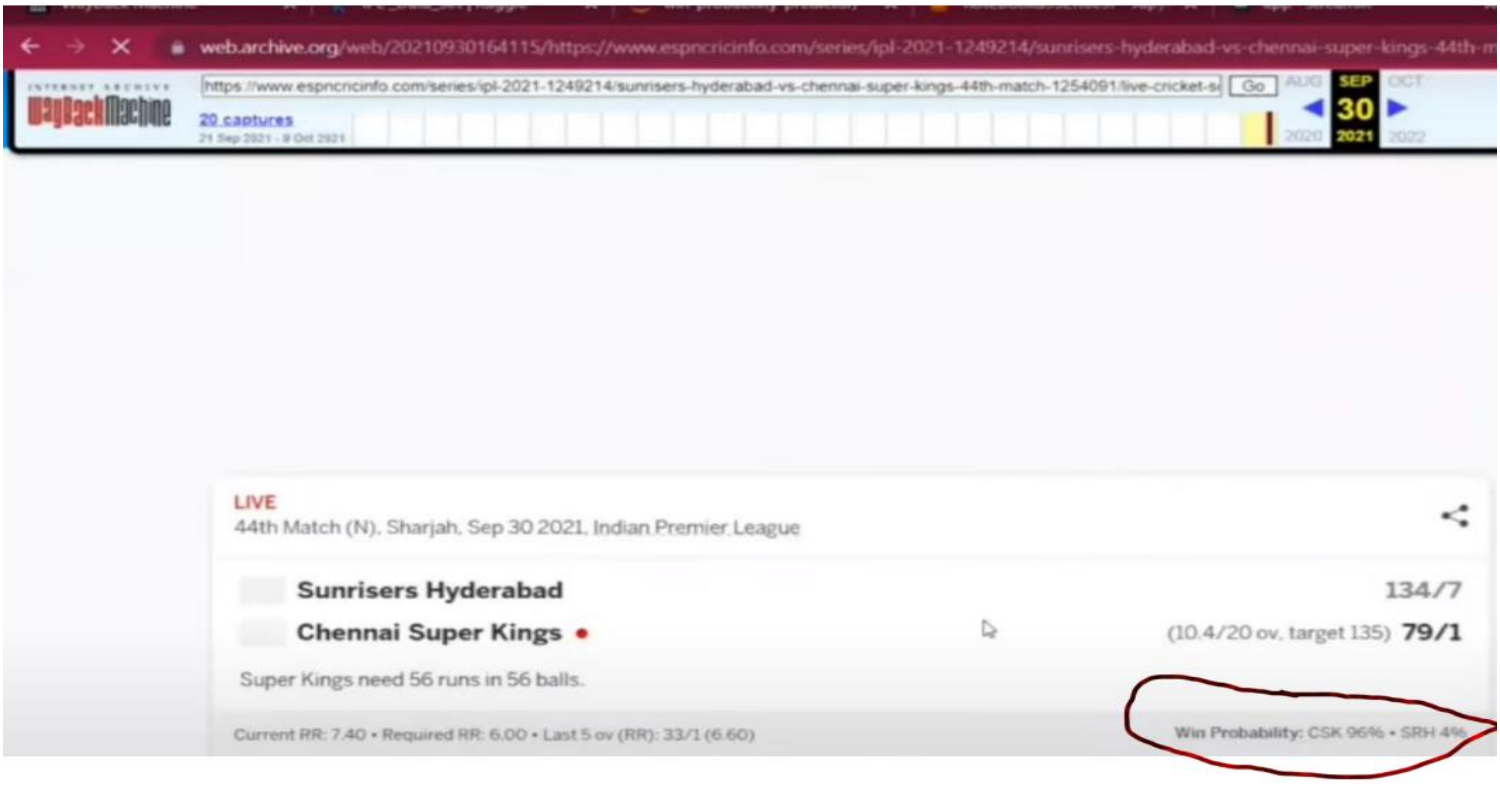
I would like to express my gratitude to **Respected Ishita Pandey**, my mentor, as well as the employee of HCLTech, for their invaluable recommendations, innovative criticisms, and support throughout the writing process. I would like to offer my heartfelt gratitude to the entire members of HCLTech.

My heartfelt gratitude goes out to my family and friends, as well as my esteemed HCL, for providing me with the chance and infrastructure to complete this project. Finally, I want to express my gratitude to everyone who assisted me in project during the creation of the project, without whom it would not have been possible.

Introduction and Idea behind creating this project:

During my session, I get to know that the HCLTech is currently working Cricket Australia, So I decided to create a project on cricket.

But what to make?? Then I recall ESPN entertainment end sports programming network which presents live match scores with winning probability for both teams in percentage, So all over the world there



are many cricket which fans who use that when it comes to know they are favourite team is winning.

Problem statement

Make a predictive system for cricket enthusiast that can predict the winning probability for the both teams.

Purpose

cricket fan love use it because they can know how much percent chances of winning team which they are supporting too.

This will increase the traffic on web and also increase the number of cricket fans which increase Net worth.

Solution

I created Indian Premier League model using machine learning algorithm such as SVM and logistic regression to present result in percentage format because it is a kind of classification problem.

During watching IPL cricket enthusiastic can input few things like falling and batting team wicket and target run to make do you know approximately how much percent chances opening the team which he/she is supporting

Technology used

→**Machine learning algorithms** are used like SVM, Logistic regression and classification to make prediction in percentage, Intelligence of system if match is having in Bangalore and Bangalore team is playing with Hyderabad so you will be observing the percentage will be bit higher than the Hyderabad.

→**Google Collaboratory** in which, I done training of model because it allows you to write codes in cells and run them independently and check the errors and output the control next to each cell.

→**Python** which is the programming language in which coding done.

- Python is a computer programming language often used to build websites and software, automate tasks.
- object-oriented, high-level programming language.

→**Logistic regression** is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient

environment for productive Python, web, and data science development. Done coding creating user interface.

Streamlit is an open-source app framework in python language. It helps us create beautiful web apps for data science and machine learning in a little time. It is compatible with major python libraries such as scikit-learn, keras, PyTorch, latex, numpy, pandas, matplotlib, etc.

Used functions like button, title etc.

Future scope

- Deploying some good website like ESPN.
- Integrating with applications.
- Work on accuracy.

Limitation

The user interface could be have more buttons & tabs to add more functionality (login, about, contact us & votes by fans to there teams with comments etc) with UI The thing is judge.

Methodology

Data Collection : The project's first step and a vital module is data collection. The Data is collected from Kaggle . From there 2 csv files were taken , one named

Indian Premier League system that predicts winning probability.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import numpy as np
import pandas as pd
```

```
match = pd.read_csv('matches.csv')
delivery = pd.read_csv('deliveries.csv')
```

```
match.head()
```

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yu
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	S
2	3	IPL-2017	Rajkot	07-04-2017	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	
3	4	IPL-2017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	C
4	5	IPL-2017	Bangalore	08-04-2017	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	H

```
delivery.head()
```

match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	...	bye_runs	legbye_runs	noball_runs
1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	...	0	0	0
1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	...	0	0	0
1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	0	...	0	0	0
1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	0	...	0	0	0
1	1	Sunrisers Hyderabad	Royal Challengers	1	5	DA Warner	S Dhawan	TS Mills	0	...	0	0	0

```
In [5]:  match.shape
Out[5]: (756, 18)

In [6]:  delivery.shape
Out[6]: (179078, 21)
```

"matches.csv" that contains IPL match data from 2008 to 2022 , like season , city, date, team1, team2, result, win_by_run, win_by_wickets, etc and the other file

Feature extraction

The data is too big to make it usable we done attribute selection.

batting_team	bowling_team	city	runs_left	balls_left	wickets_left	total_runs_x	crr	rrr	result
Kings XI Punjab	Rajasthan Royals	Durban	208	108	8	211	1.500000	11.555556	0
Royal Challengers Bangalore	Mumbai Indians	Mumbai	91	66	9	141	5.555556	8.272727	1
Kolkata Knight Riders	Delhi Daredevils	Delhi	97	83	8	146	7.945946	7.012048	1
Mumbai Indians	Chennai Super Kings	Mumbai	69	34	6	168	6.906977	12.176471	0
Rajasthan Royals	Chennai Super Kings	Chennai	113	81	9	157	6.769231	8.370370	0

rrr= required run rate & crr= current run rate.

Data pre-processing : To produce results that are incredibly accurate and insightful, machine learning significantly relies on data pre-processing. The data quality is better the more trustworthy the findings that are provided. Data from real-world datasets tends to be sparse, noisy, and unreliable. By completing gaps in the data, eliminating noise, and

Data Pre-processing

```
total_score_df = delivery.groupby(['match_id','inning']).sum()['total_runs'].reset_index()
```

```
total_score_df
```

7]:

	match_id	inning	total_runs
0	1	1	207
1	1	2	172
2	2	1	184
3	2	2	187
4	3	1	183
...	...	...	...
1523	11413	2	170
1524	11414	1	155
1525	11414	2	162
1526	11415	1	152
1527	11415	2	157

addressing anomalies, data pre-processing enhances the quality of the data. Here we took the important data from deliveries.csv and **created a new data frame named** "total_score_df" which includes "match_id", "inning" and also introduced a new column of "total_runs" for each match and inning.

From above Data Frame we processed it and took out the score of only 1st inning of every match.

After that we have merged the "matches.csv" data frame with our newly built data frame i.e. "total_score_df"

.reset_index() used to create data frame

filtering the data for first inning, Now merging with match data set.

```
match_df = match.merge(total_score_df[['match_id','total_runs']], left_on='id',right_on='match_id')
```

```
match_df
```

n	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_match	venue	umpire1	umpire2	umpire3	match_id	total_runs
d	normal	0	Sunrisers Hyderabad	35	0	Yuvraj Singh	Rajiv Gandhi International Stadium, Uppal	AY Dandekar	NJ Llong	NaN	1	207
d	normal	0	Rising Pune Supergiant	0	7	SPD Smith	Maharashtra Cricket Association Stadium	A Nand Kishore	S Ravi	NaN	2	184
d	normal	0	Kolkata Knight Riders	0	10	CA Lynn	Saurashtra Cricket Association Stadium	Nitin Menon	CK Nandan	NaN	3	183
d	normal	0	Kings XI Punjab	0	6	GJ Maxwell	Holkar Cricket Stadium	AK Chaudhary	C Shamshuddin	NaN	4	163
it	normal	0	Royal Challengers Bangalore	15	0	KM Jadhav	M Chinnaswamy Stadium	NaN	NaN	NaN	5	157
...	...	...	...	...	...	...	...	...	...	...	...	...
d	normal	0	Mumbai Indians	0	9	HH Pandya	Wankhede Stadium	Nanda Kishore	O Nandan	S Ravi	11347	143
it	normal	0	Mumbai Indians	0	6	AS Yadav	M. A. Chidambaram Stadium	Nigel Llong	Nitin Menon	Ian Gould	11412	136

Because we want match ID and total runs not in innings, join condition to get total runs in first inning .

Data Cleaning: Data cleaning is the process of removing mistakes and substituting real values for them. The collected data sets need to be cleaned because they contain noisy data like null values and unsuitable values. So that we can adequately evaluate the data, the

```
match_df['team1'].unique()
```

```
array(['Sunrisers Hyderabad', 'Mumbai Indians', 'Gujarat Lions',
       'Rising Pune Supergiant', 'Royal Challengers Bangalore',
       'Kolkata Knight Riders', 'Delhi Daredevils', 'Kings XI Punjab',
       'Chennai Super Kings', 'Rajasthan Royals', 'Deccan Chargers',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Rising Pune Supergiants',
       'Delhi Capitals'], dtype=object)
```

data is cleaned by replacing null values with zeros and grouped into the appropriate columns.

Lot of team participates but few of them not play like Pune Gujarat Karla we need to keep only those team who played.

```
teams = [
    'Sunrisers Hyderabad',
    'Mumbai Indians',
    'Royal Challengers Bangalore',
    'Kolkata Knight Riders',
    'Kings XI Punjab',
    'Chennai Super Kings',
    'Rajasthan Royals',
    'Delhi Capitals'
]
```

```
match_df['team1'] = match_df['team1'].str.replace('Delhi Daredevils','Delhi Capitals')
match_df['team2'] = match_df['team2'].str.replace('Delhi Daredevils','Delhi Capitals')

match_df['team1'] = match_df['team1'].str.replace('Deccan Chargers','Sunrisers Hyderabad')
match_df['team2'] = match_df['team2'].str.replace('Deccan Chargers','Sunrisers Hyderabad')
```

```
[67] match_df = match_df[match_df['team1'].isin(teams)]
     match_df = match_df[match_df['team2'].isin(teams)]
```

Replacing 2 team names which are same but different name.

Now keeping only those teams which are playing & which are present in teams variable.

```
[68] match_df.shape

     (641, 20)
```

755	11415	IPL-2019	Hyderab

756 rows × 20 columns

Now only 641 from 756 after **data cleaning.**

```
[120] match_df = match_df[match_df['dl_applied'] == 0]

[121] match_df = match_df[['match_id','city','winner','total_runs']]

[122] delivery_df = match_df.merge(delivery,on='match_id')

[123] delivery_df = delivery_df[delivery_df['inning'] == 2]

[124] delivery_df
```

di_applied shows rain affected matches but we want the matches which not affected by rain.

```
delivery_df
```

nning	batting_team	bowling_team	over	ball	batsman	...	bye_runs	legbye_runs	noball_runs	penalty_runs	batsman_runs	extra_runs	total_runs_y	player_dismissed	dismissal
2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	1	CH Gayle	...	0	0	0	0	1	0	1	NaN	
2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	2	Mandeep Singh	...	0	0	0	0	0	0	0	NaN	
2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	3	Mandeep Singh	...	0	0	0	0	0	0	0	NaN	
2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	4	Mandeep Singh	...	0	0	0	0	2	0	2	NaN	
2	Royal Challengers	Sunrisers	1	5	Mandeep		0	0	0	0	1	0	1	NaN	

```
[125] delivery_df['current_score'] = delivery_df.groupby('match_id').cumsum()['total_runs_y']

<ipython-input-125-cafdf4636499>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.cumsum is deprecated. In a future version, numeric_only will default to
  delivery_df['current_score'] = delivery_df.groupby('match_id').cumsum()['total_runs_y']

[126] delivery_df['runs_left'] = delivery_df['total_runs_x'] - delivery_df['current_score']
```

Taking column and merging with delivery data

need to find run left, ball left, wicket left and result so totalruns_y shows run on each ball.

Getting cumulative sum means some after each ball new attribute created current score then total run - current score gives **run left attribute value**.

```
delivery_df['balls_left'] = 126 - (delivery_df['over']*6 + delivery_df['ball'])

delivery_df
```

m	bowling_team	over	ball	batsman	...	penalty_runs	batsman_runs	extra_runs	total_runs_y	player_dismissed	dismissal_kind	fielder	current_score	runs_left	balls_left
al s e	Sunrisers Hyderabad	1	1	CH Gayle	...	0	1	0	1	NaN	NaN	NaN	1	206	119

Created ball left attribute by the help of over and balls field.

```
delivery_df['player_dismissed'] = delivery_df['player_dismissed'].fillna("0")
delivery_df['player_dismissed'] = delivery_df['player_dismissed'].apply(lambda x:x if x == "0" else "1")
delivery_df['player_dismissed'] = delivery_df['player_dismissed'].astype('int')
wickets = delivery_df.groupby('match_id').cumsum()['player_dismissed'].values
delivery_df['wickets'] = 10 - wickets
delivery_df.head()

<ipython-input-129-5cbb94c1e6d2>:4: FutureWarning: The default value of numeric_only in DataFrameGroupBy.c
```

To get wicket left will you use attribute called player dismissed

fifillna() function used to replace NaN to 0

.astype used to convert to integer

now we know wicket left after each ball

```
delivery_df.tail()
```

ng_team	bowling_team	over	ball	batsman	...	batsman_runs	extra_runs	total_runs_y	player_dismissed	dismissal_kind	fielder	current_score	runs_left	balls_left	wickets
Chennai er Kings	Mumbai Indians	20	2	RA Jadeja	...	1	0	1	0	NaN	NaN	152	0	4	5
Chennai er Kings	Mumbai Indians	20	3	SR Watson	...	2	0	2	0	NaN	NaN	154	-2	3	5
Chennai er Kings	Mumbai Indians	20	4	SR Watson	...	1	0	1	1	run out	KH Pandya	155	-3	2	4
Chennai	Mumbai	20	5	SN		2	0	2	0	NaN	NaN	157	5	1	4

```
[34] # crr = runs/overs
     delivery_df['crr'] = (delivery_df['current_score']*6)/(120 - delivery_df['balls_left'])

[35] delivery_df['rrr'] = (delivery_df['runs_left']*6)/delivery_df['balls_left']
```

example 1 * 6 is equals to 6 / 2 so run rate will be 3.0

```
def result(row):
    return 1 if row['batting_team'] == row['winner'] else 0

[37] delivery_df['result'] = delivery_df.apply(result,axis=1)

[38] final_df = delivery_df[['batting_team','bowling_team','city','runs_left','balls_left','wickets','total_runs_x','crr','rrr','result']]
```

required run rate equals to run * 6 divided by ball left

to get result column make a function if winning and batting team it same so make it one use apply function.

Taking attributes in desired order.

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

[40] final_df.sample()
```

	batting_team	bowling_team	city	runs_left	balls_left	wickets	total_runs_x	crr	rrr	result
9593	Rajasthan Royals	Royal Challengers Bangalore	Bangalore	72	70	8	135	7.56	6.171429	1

```
[41] final_df.dropna(inplace=True)

[42] final_df = final_df[final_df['balls_left'] != 0]

X = final_df.iloc[:,:-1]
y = final_df.iloc[:,-1]
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)
```

Now model building splitting into training and test iloc is all rows except last

```
X_train
```

	batting_team	bowling_team	city	runs_left	balls_left	wickets	total_runs_x	crr	rrr
147387	Sunrisers Hyderabad	Mumbai Indians	Mumbai	71	42	7	170	7.615385	10.142857
23265	Kolkata Knight Riders	Mumbai Indians	Port Elizabeth	168	98	8	187	5.181818	10.285714
89878	Rajasthan Royals	Royal Challengers Bangalore	Abu Dhabi	-1	42	6	70	5.461538	-0.142857
48934	Rajasthan Royals	Delhi Daredevils	Jaipur	93	80	9	151	8.700000	6.975000

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

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

```
[46] from sklearn.linear_model import LogisticRegression
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.pipeline import Pipeline
```

```
[47] pipe = Pipeline(steps=[
         ('step1',trf),
         ('step2',LogisticRegression(solver='liblinear'))
     ])
```

3 fields are strength so we have to use **one hot encoding** and making **pipeline** step one transformation and Step 2 logistic regression model.

One error which I encountered 800 plus missing value in one field.

```
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').
```

```
In [101]: final_df.isnull().sum()

Out[101]: batting_team      0
          bowling_team      0
          city            832
          runs_left         0
          balls_left        0
          wickets           0
```

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

by drop

```
[48] pipe.fit(X_train,y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_outp
  warnings.warn(
Pipeline
step1: ColumnTransformer
trf
remainder
OneHotEncoder
passthrough
LogisticRegression

[49] y_pred = pipe.predict(X_test)

[50] from sklearn.metrics import accuracy_score
     accuracy_score(y_test,y_pred)

0.8036302473894457
```

after training getting accuracy 80 which shows quality of the model no problem with under fitting and over fitting.

Logistic regression algorithm is used conversely (but) earlier, I tried to use **random forest** algorithm but it increased the training time & **accuracy** was **99** but we want to know at each stage for both team winning probability so logistic regression, I found better.

```
[48] def match_progression(x_df,match_id,pipe):
        match = x_df[x_df['match_id'] == match_id]
        match = match[(match['ball'] == 6)]
        temp_df = match[['batting_team','bowling_team','city','runs_left','balls_left','wickets','total_runs_x','crr','rrr']].dropna()
        temp_df = temp_df[temp_df['balls_left'] != 0]
        result = pipe.predict_proba(temp_df)
        temp_df['lose'] = np.round(result.T[0]*100,1)
        temp_df['win'] = np.round(result.T[1]*100,1)
        temp_df['end_of_over'] = range(1,temp_df.shape[0]+1)

        target = temp_df['total_runs_x'].values[0]
        runs = list(temp_df['runs_left'].values)
        new_runs = runs[:]
        runs.insert(0,target)
        temp_df['runs_after_over'] = np.array(runs)[:-1] - np.array(new_runs)
        wickets = list(temp_df['wickets'].values)
        new_wickets = wickets[:]
        new_wickets.insert(0,10)
        wickets.append(0)
        w = np.array(wickets)
        nw = np.array(new_wickets)
        temp_df['wickets_in_over'] = (nw - w)[0:temp_df.shape[0]]

        print("Target-",target)
        temp_df = temp_df[['end_of_over','runs_after_over','wickets_in_over','lose','win']]
        return temp_df,target
```

If we give complete ball by ball data and give one particular match so it will show over by over probability.

Taking all match rows and sending to pipe and it will give result.

```
temp_df,target = match_progression(delivery_df,74,pipe)
temp_df
```

```
Target- 178
```

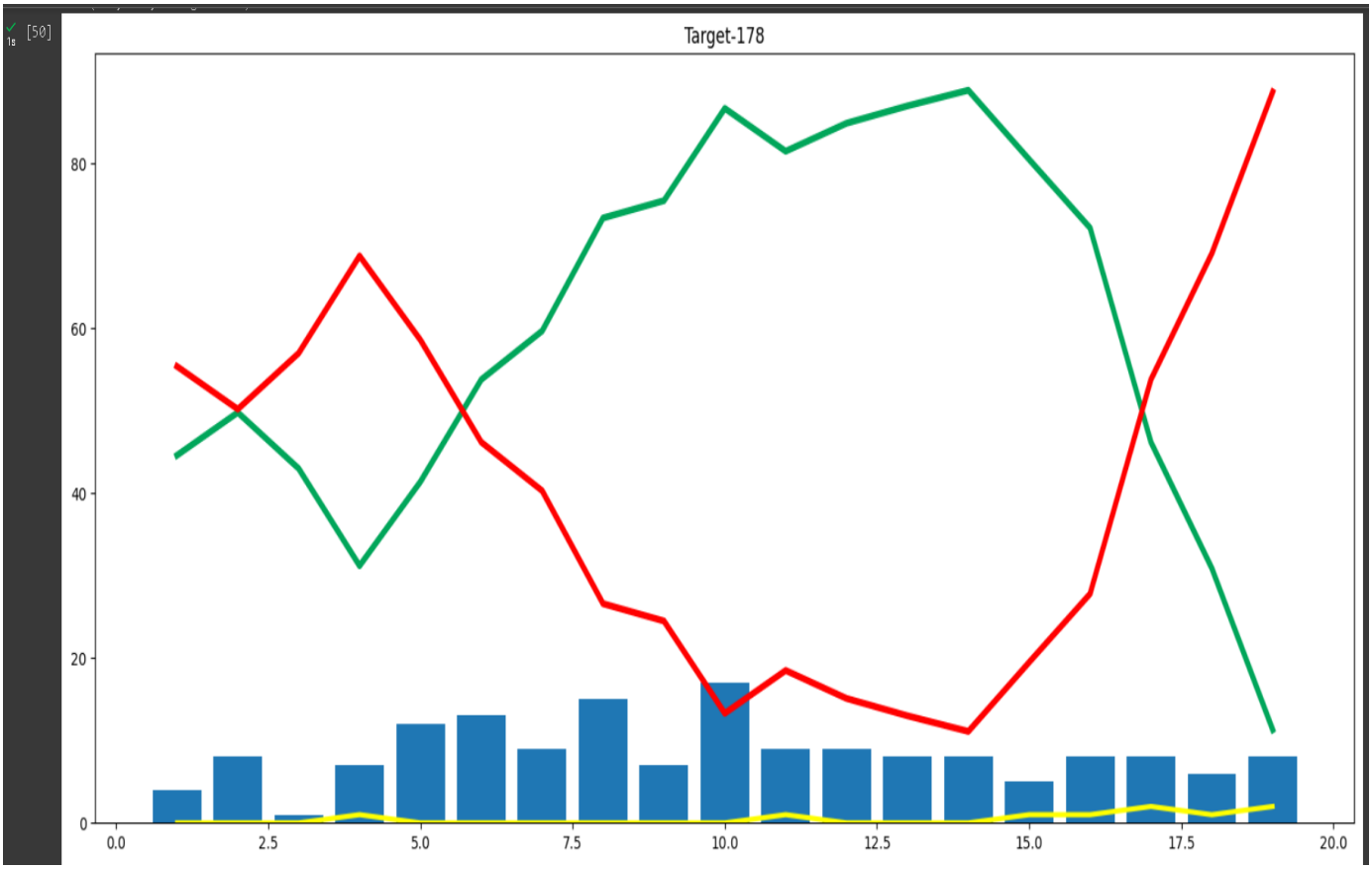
So for this match after first over there is 4 runs and target is 178 so losing probability is 55 and winning is 44.

```
Target- 178
```

	end_of_over	runs_after_over	wickets_in_over	lose	win
10459	1	4	0	55.4	44.6
10467	2	8	0	50.2	49.8
10473	3	1	0	57.0	43.0
10479	4	7	1	68.8	31.2
10485	5	12	0	58.6	41.4
10491	6	13	0	46.2	53.8
10497	7	9	0	40.3	59.7
10505	8	15	0	26.6	73.4
10511	9	7	0	24.5	75.5
10518	10	17	0	13.3	86.7
10524	11	9	1	18.5	81.5
10530	12	9	0	15.1	84.9
10536	13	8	0	13.0	87.0
10542	14	8	0	11.1	88.9
10548	15	5	1	19.5	80.5
10555	16	8	1	27.8	72.2
10561	17	8	2	53.8	46.2
10567	18	6	1	69.1	30.9

```
import matplotlib.pyplot as plt
plt.figure(figsize=(18,8))
plt.plot(temp_df['end_of_over'],temp_df['wickets_in_over'],color='yellow',linewidth=3)
plt.plot(temp_df['end_of_over'],temp_df['win'],color='#00a65a',linewidth=4)
plt.plot(temp_df['end_of_over'],temp_df['lose'],color='red',linewidth=4)
plt.bar(temp_df['end_of_over'],temp_df['runs_after_over'])
plt.title('Target-' + str(target))
```

```
Text(0.5, 1.0, 'Target-178')
```



In this match sometime one team is going ahead and another back and visa versa.

But after 17th over decidable flow came.

we can also study about matches using this model.

Deployment

required 3 things

all team names

```
[51] teams

['Sunrisers Hyderabad',
 'Mumbai Indians',
 'Royal Challengers Bangalore',
 'Kolkata Knight Riders',
 'Kings XI Punjab',
 'Chennai Super Kings',
 'Rajasthan Royals',
 'Delhi Capitals']
```

second city names

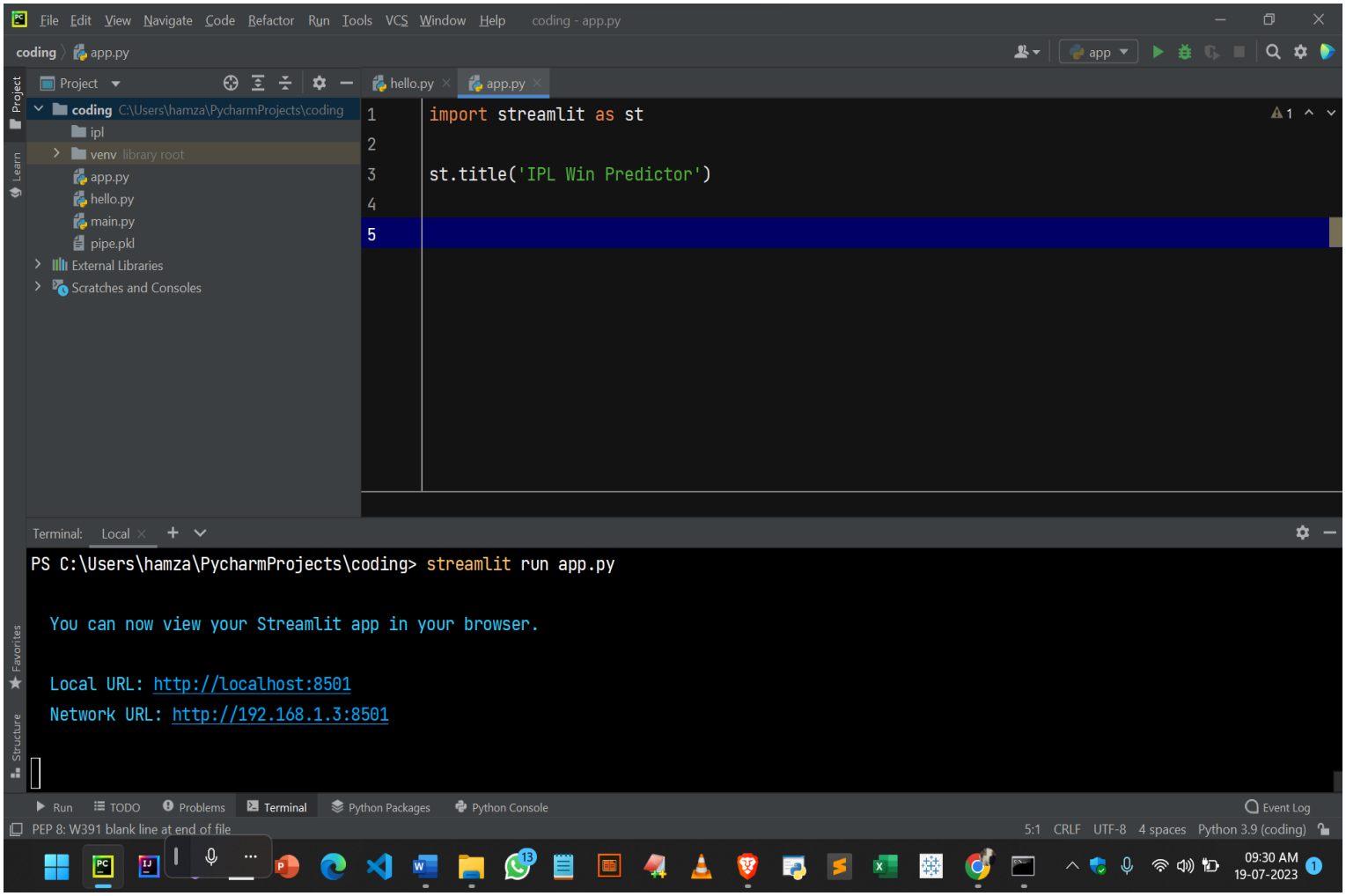
```
[52] delivery_df['city'].unique()

array(['Hyderabad', 'Bangalore', 'Mumbai', 'Indore', 'Kolkata', 'Delhi',
       'Chandigarh', 'Jaipur', 'Chennai', 'Cape Town', 'Port Elizabeth',
       'Durban', 'Centurion', 'East London', 'Johannesburg', 'Kimberley',
       'Bloemfontein', 'Ahmedabad', 'Cuttack', 'Nagpur', 'Dharamsala',
       'Visakhapatnam', 'Pune', 'Raipur', 'Ranchi', 'Abu Dhabi',
       'Sharjah', nan, 'Mohali', 'Bengaluru'], dtype=object)
```

3rd create pipe object.

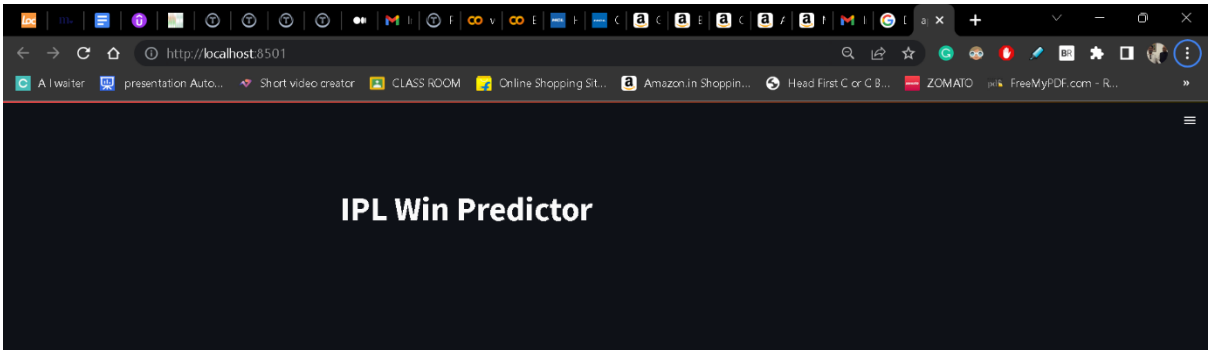
```
[53] import pickle
     pickle.dump(pipe,open('pipe.pkl','wb'))
```

Created a folder in PC and also pasted pipe.pkl file

install streamlit and updated pip version.

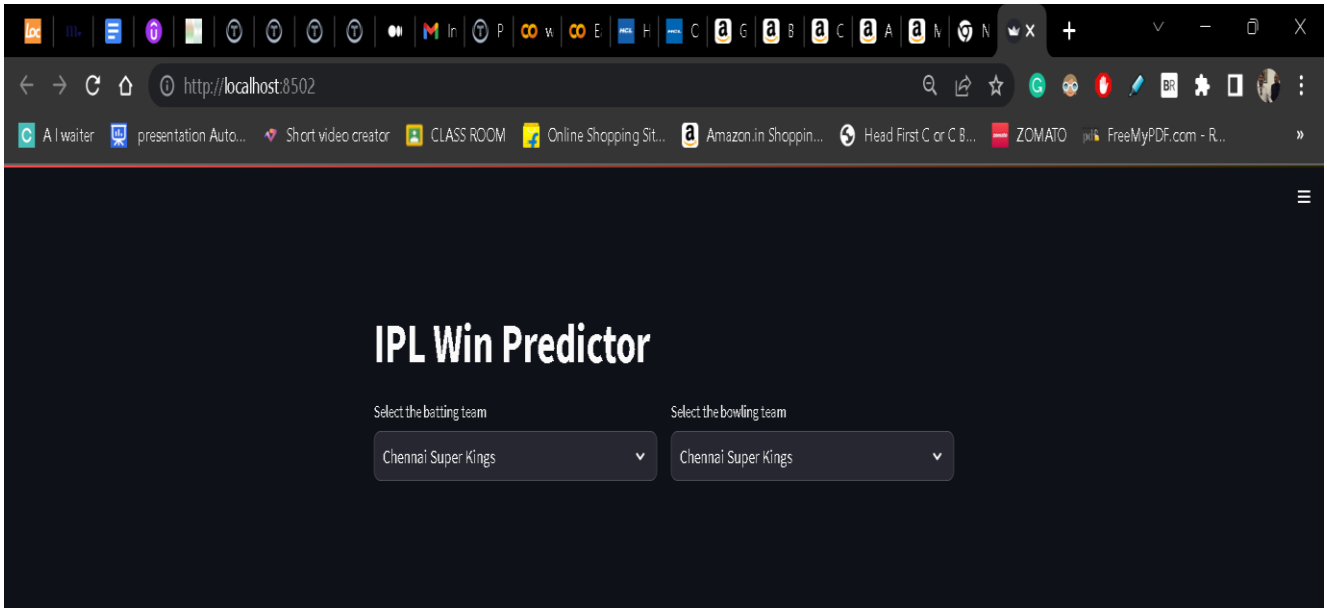


On Chrome browser.

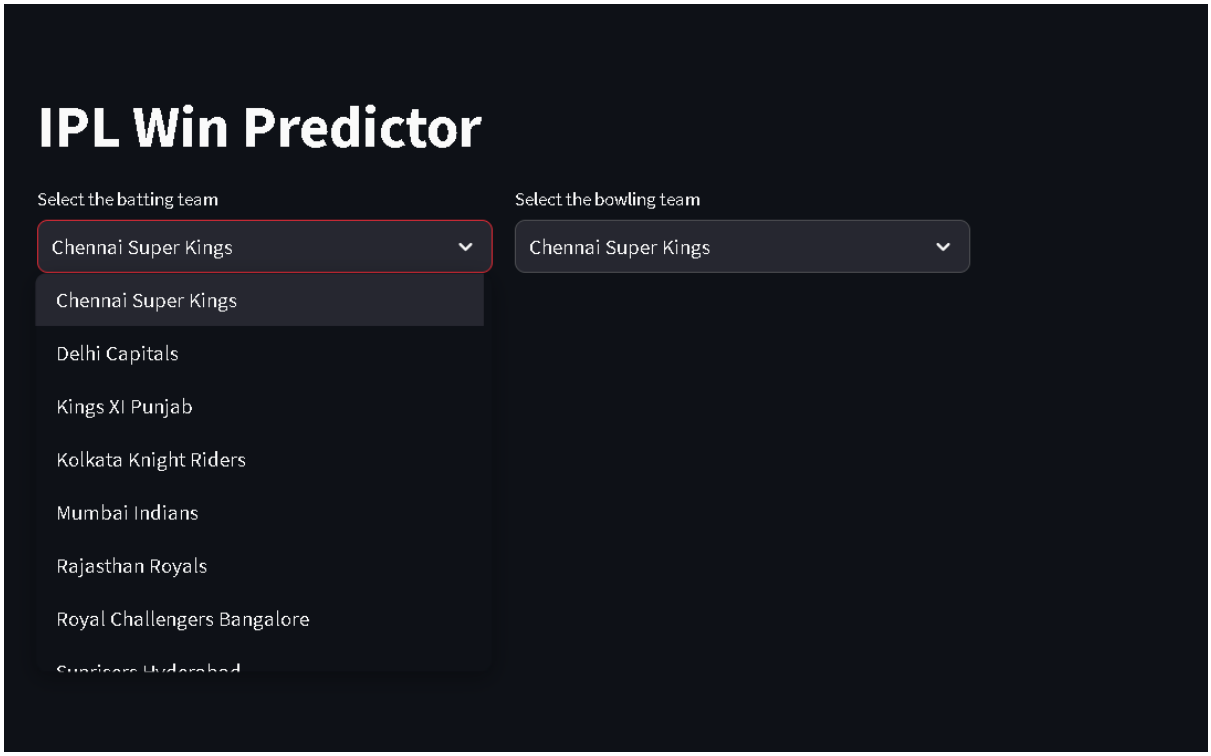


```python
pipe = pickle.load(open('pipe.pkl','rb'))
st.title('IPL Win Predictor')

col1, col2 = st.columns(2)

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

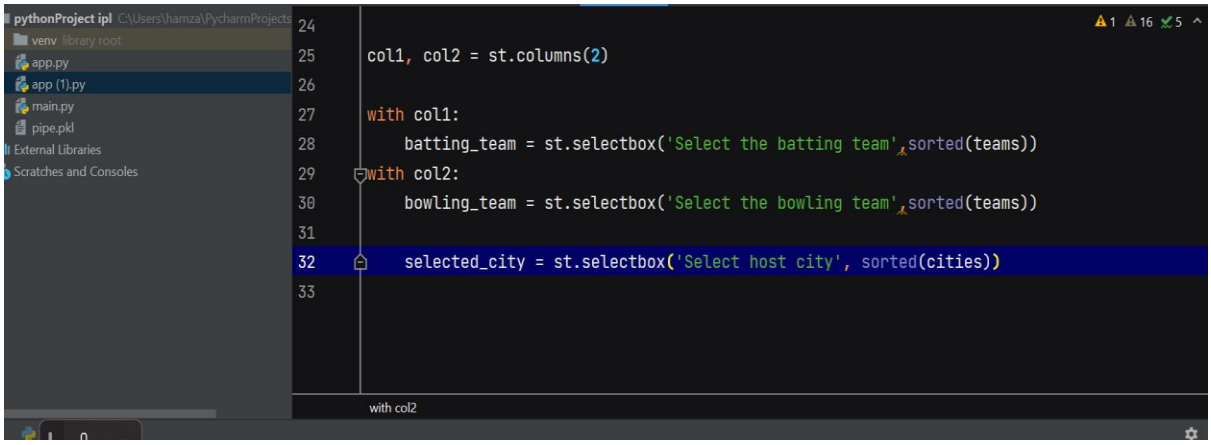
Output of above code used column function



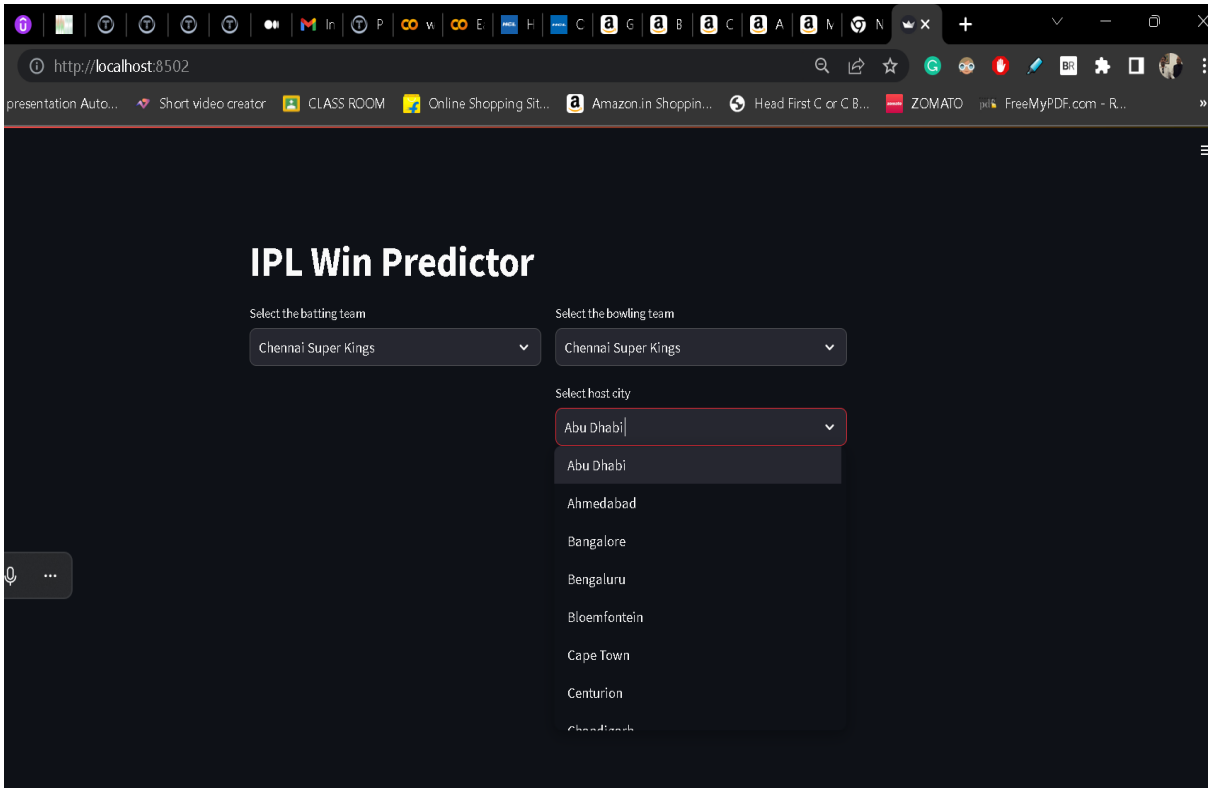
drop down box created with select box function



now creating drop box to select host city



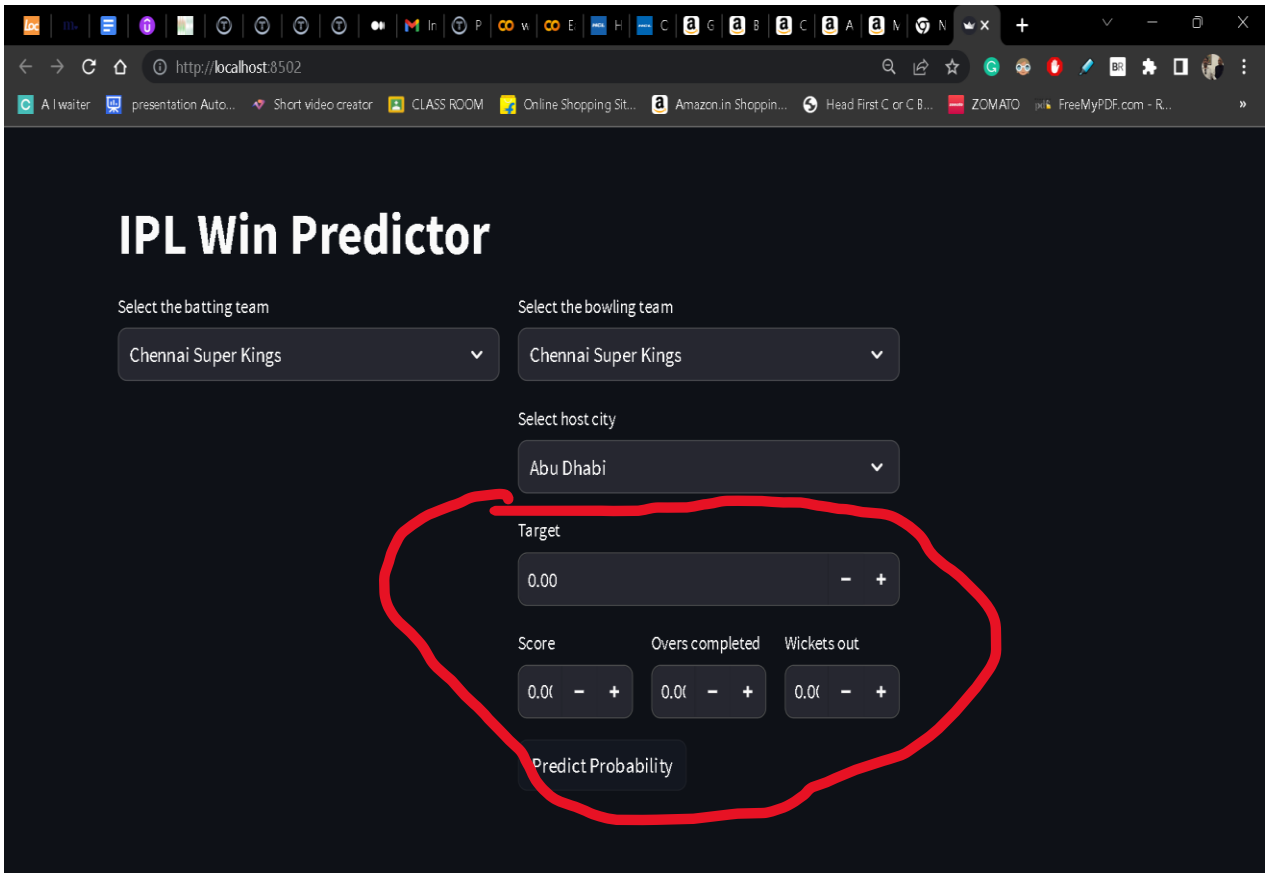
sorted function use for alphabetically sorting



now writing code to make field which can take *target* from the user and 3 columns for **wicket left, overs completed & score.**

```
target = st.number_input('Target')

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

with col3:
    score = st.number_input('Score')
with col4:
    overs = st.number_input('Overs completed')
with col5:
```

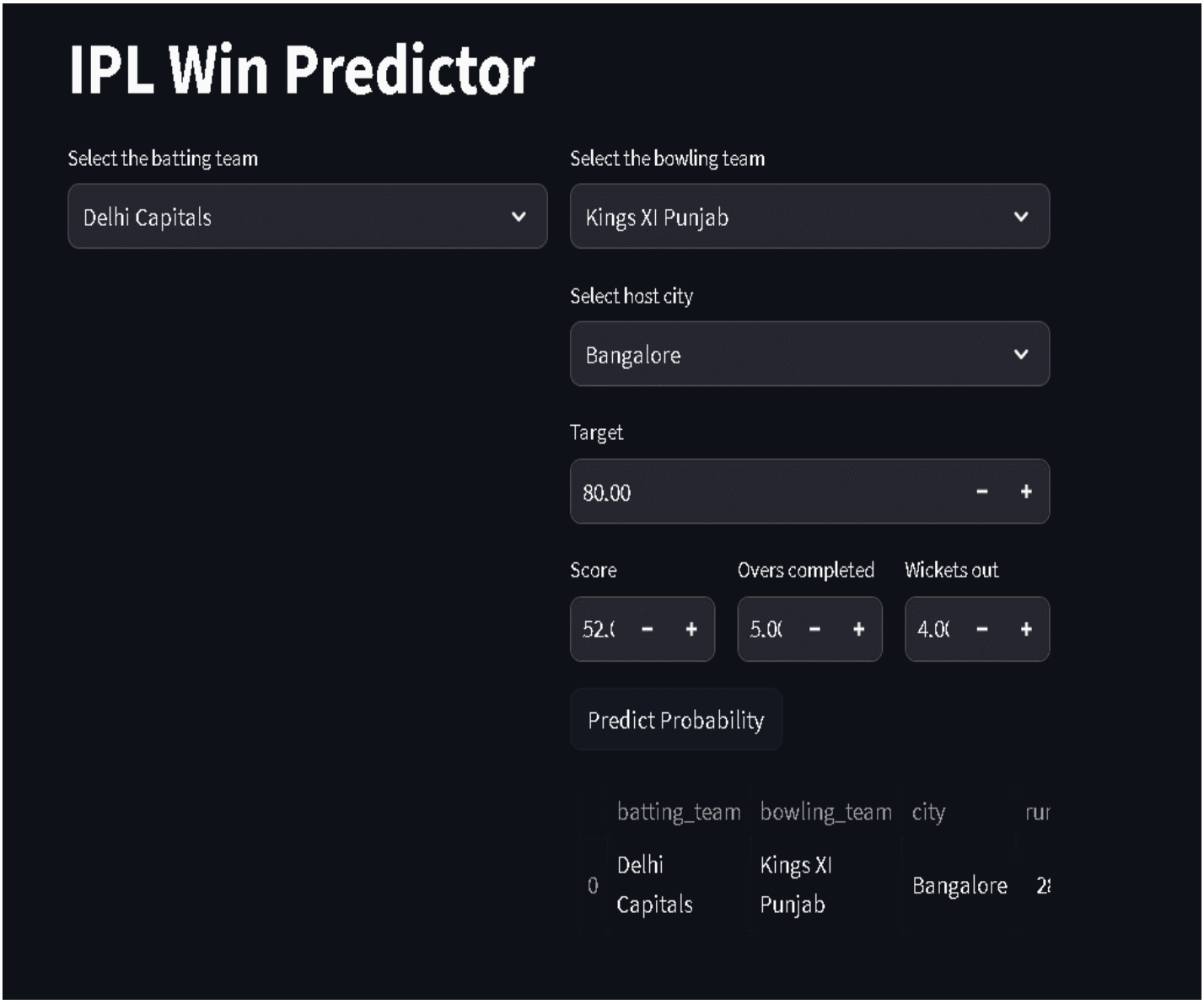
```
wickets = st.number_input('Wickets out')

if st.button('Predict Probability'):
```



let's write formulas for these

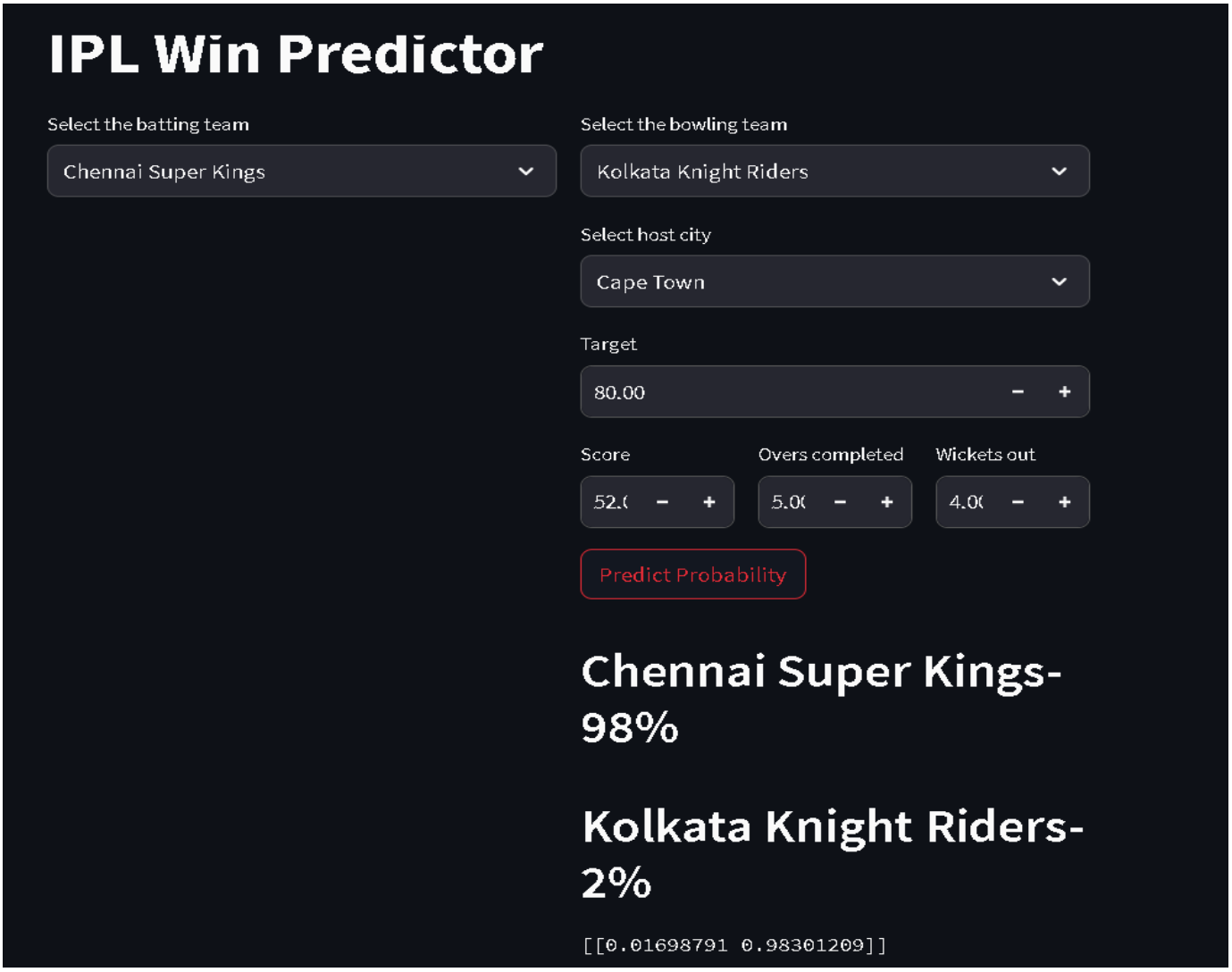
```
runs_left = target - score
balls_left = 120 - (overs * 6)
wickets = 10 - wickets
crr = score / overs
rrr = (runs_left * 6) / balls_left
```

using pandas in table frame() result



	batting_team	bowling_team	city	runs_left	balls_left	wickets	total_runs_x	crr	rrr
0	Delhi Capitals	Kings XI Punjab	Bangalore	28.0000	90.0000	6.0000	80.0000	10.4000	1.8667

```
input_df = pd.DataFrame(
    {'batting_team': [batting_team], 'bowling_team': [bowling_team], 'city': [selecte
     'runs_left': [runs_left], 'balls_left': [balls_left], 'wickets': [wickets], 'tot
     'crr': [crr], 'rrr': [rrr]})
#st.table(input_df)
```



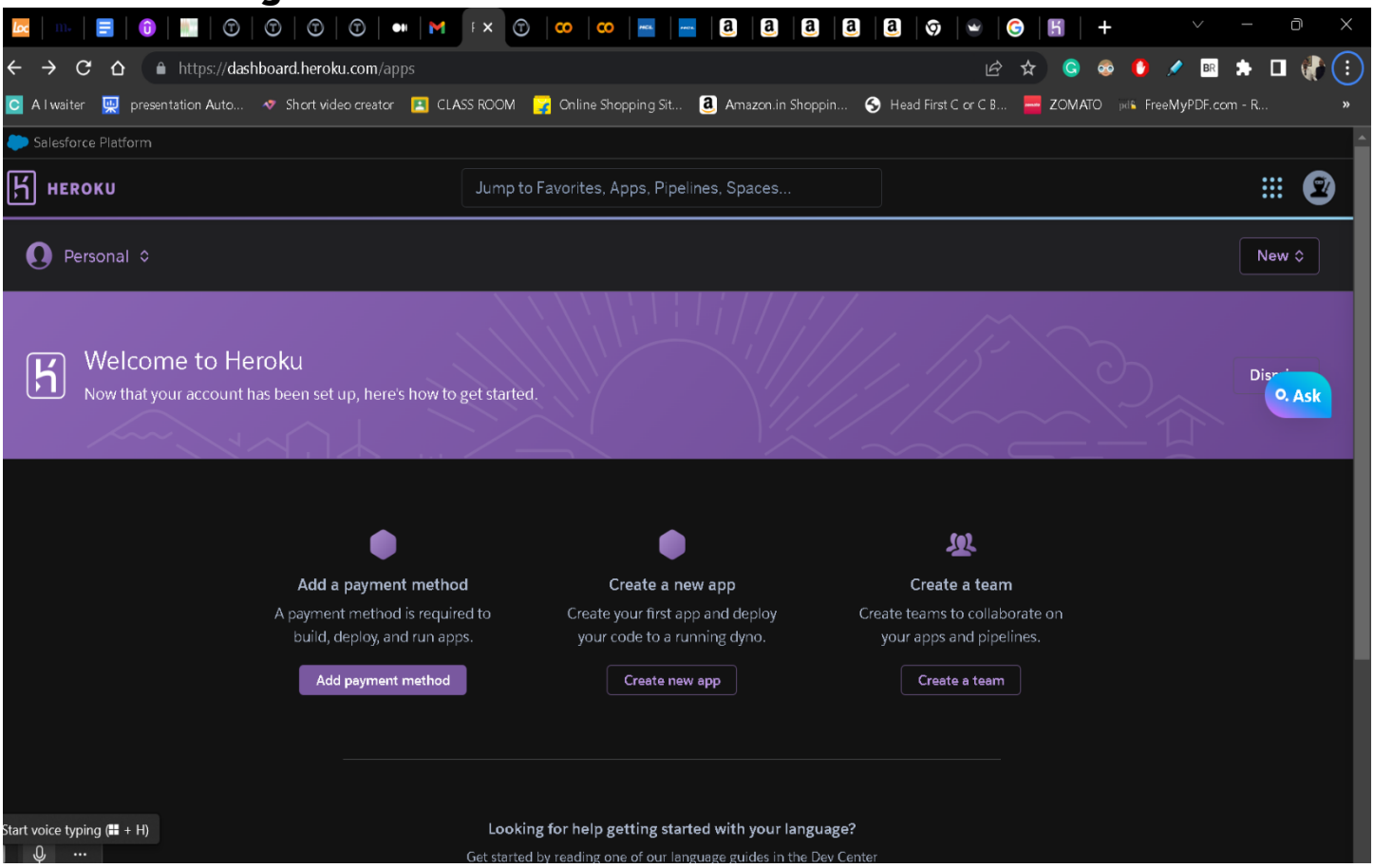

```
result = pipe.predict_proba(input_df)
loss = result[0][0]
win = result[0][1]
st.header(batting_team + "- " + str(round(win * 100)) + "%")
st.header(bowling_team + "- " + str(round(loss * 100)) + "%")

st.text(result)
```

header() make it bold and big.

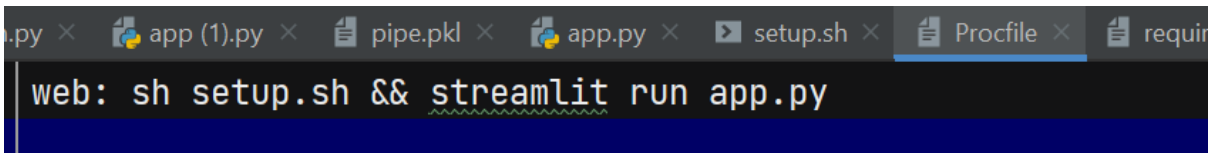
Deploy on HEROKU

Is a platform as a service that enables developers to build and run operate application entirely in the cloud.

Let's log into it



Now needs to create few files setup.sh, procfile(have command which run app) & requirement(it tells to install packages on server).



main.py app (1).py pipe.pkl app.py setup.sh Procfile requirements.txt

kage requirement 'sklearn' is not satisfied Install

```
streamlit
pandas
numpy
sklearn
```

main.py app (1).py pipe.pkl app.py setup.sh Procfile requirements.txt

```
mkdir -p ~/.streamlit/

echo "\
[server]\n\
port = $PORT\n\
enableCORS = false\n\
headless = true\n\
\n\
" > ~/.streamlit/config.toml
```

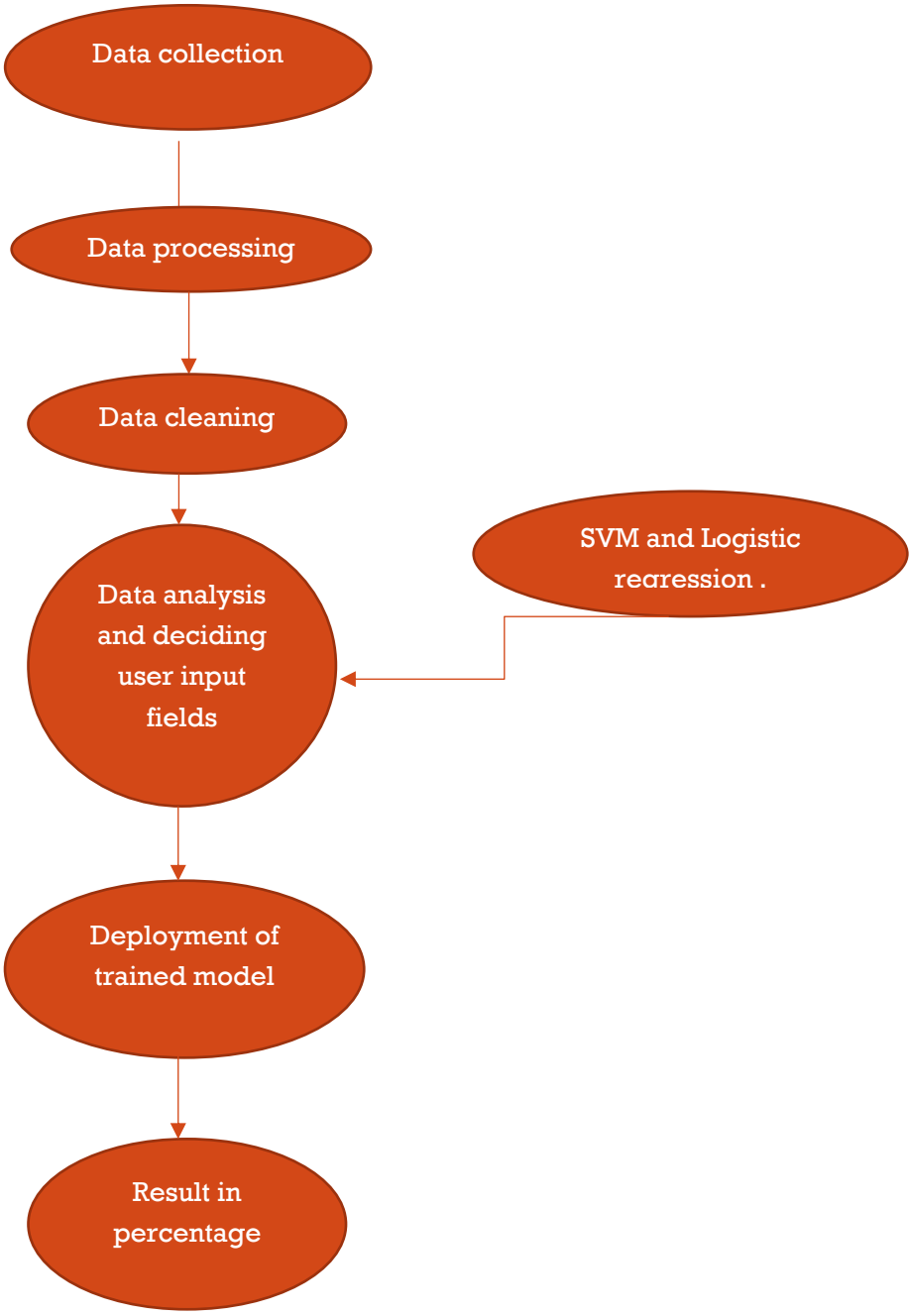
```
Users/hamza/PycharmProjects/pythonProject ipl/app.py [ARGUMENTS]
t call last):
a\PycharmProjects\pythonProject ipl\app.py", line 58, in <module>
dict_proba(input_df)
t_df' is not defined

exit code 1
```

Python Packages Python Console Event Log

ed (41 minutes ago) 10:1 LF UTF-8 2 spaces Python 3.9 (pythonProject ipl)

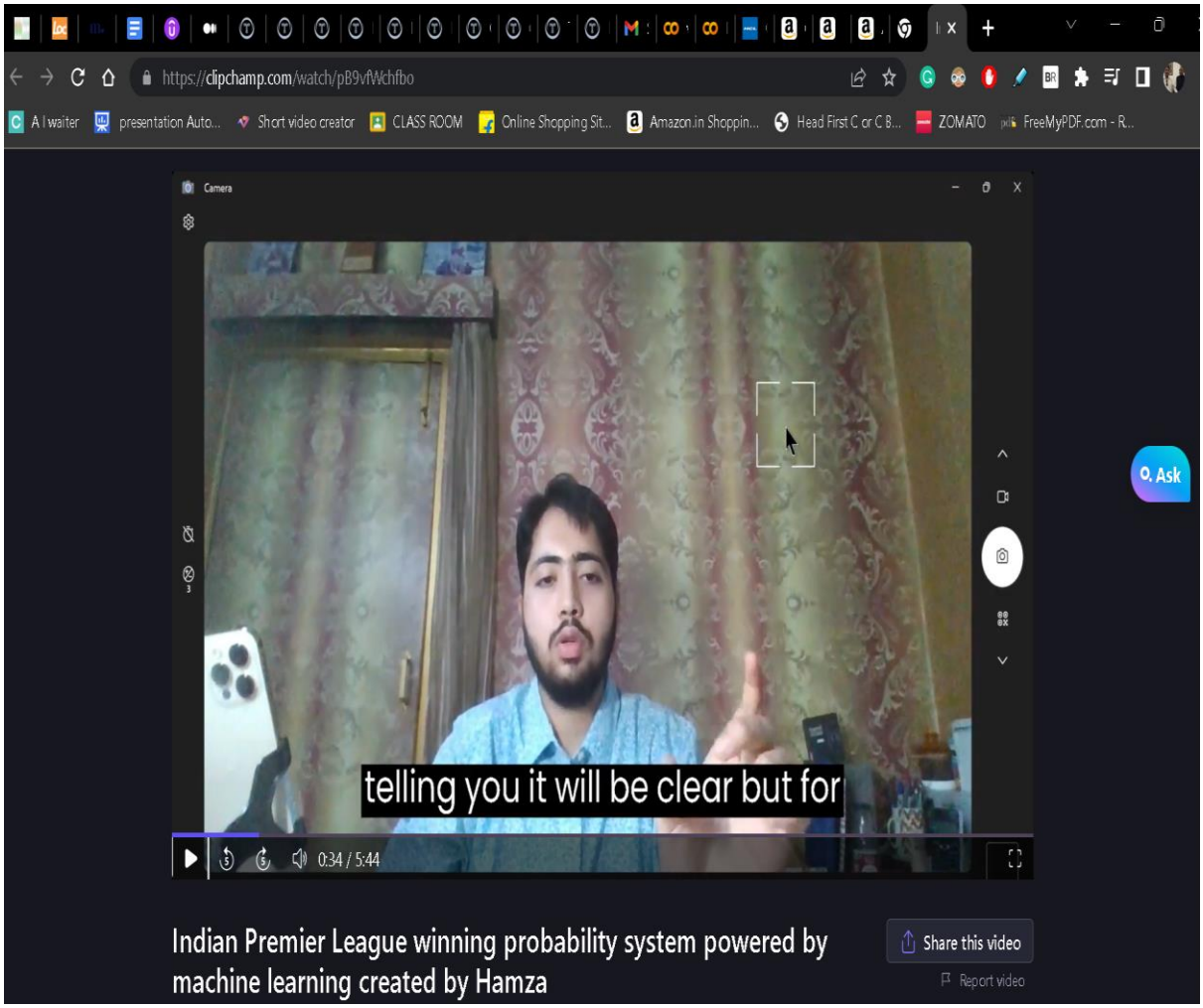
05:26 PM 19-07-2023

Data flow diagram



Clip which shows demonstration of project

click on following image to redirect to **watch online in my voice**.



conclusion

This is predictive system which, I create for cricket enthusiast by which fans can know how much percent chances their team have to win the match!!

And it is accurate **conversely** (but) cannot beat **ESPN** predictive system as but in future will be trying to take **accuracy to above 90 from 80**.

Useful links

link to watch online working of this project in my voice.

https://clipchamp.com/watch/pB9vfWchfbo

Link to download the clip.

https://drive.google.com/uc?id=16W6VlV-oqFng-5_7o9qXyHYpQMW3UJlX&export=download


Thank You!