

Project Report

Cricket Win Probability Predictor

Predicting the winning probability of an IPL Cricket team in a particular match scenario.

Submitted By:

Name: Raj Kansal

Email: rajkansal2001@gmail.com

Ph No.: 9309741329

College: D Y Patil International University, Akurdi, Pune

Branch: B.Tech 3rd Year

Abstract

In this Data Science Project, I have made a cricket win probability predictor that predicts whether a particular team would win or not given the details of a particular match situation.

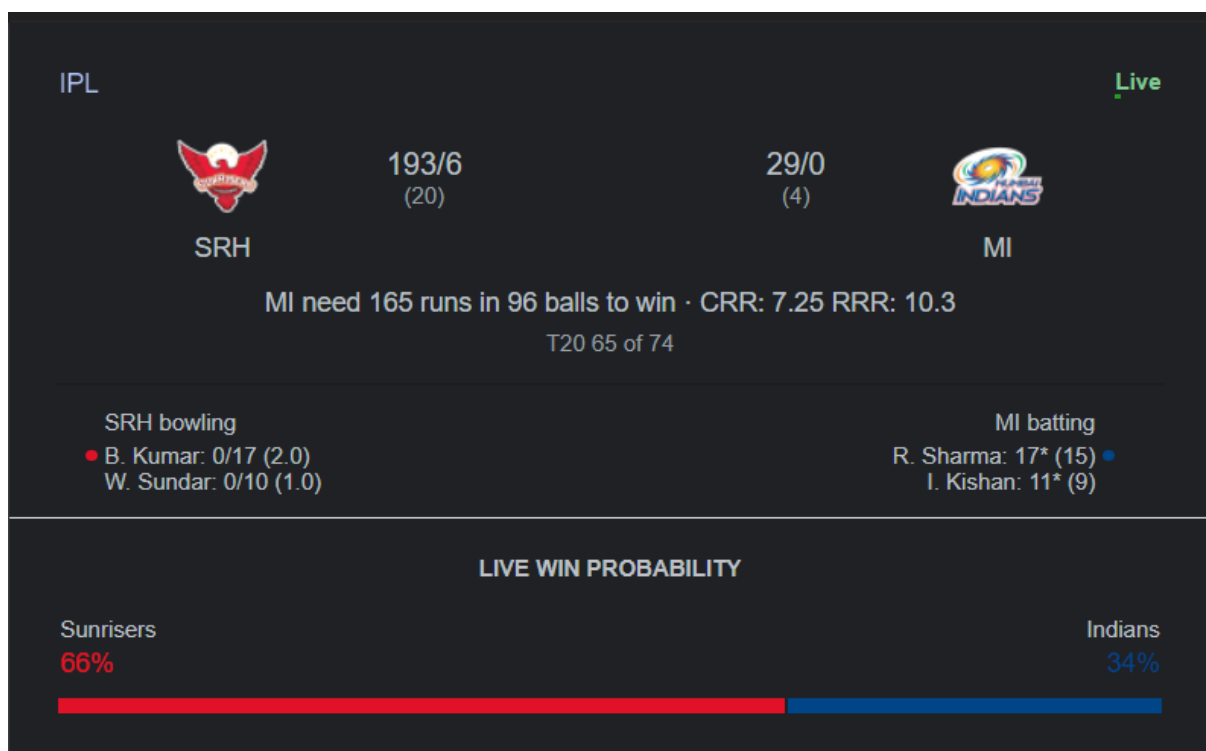
This project is based on **Classification Problem – Logistic Regression**.

Let's understand this with an example –


Suppose there's a match between Sunrisers Hyderabad (SRH) and Mumbai Indians (MI). During 1st innings SRH bats and gives a target. Now the winning probability of both the teams depends on the 2nd innings, i.e the team that will bat in 2nd innings; here MI will be batting in 2nd innings. Based on the team's performance playing the 2nd innings, we can find the win probability determining which team would win and which team would lose.

Here's an image based on Google's data that depicts the same.



(Example from Google)



Technologies Used:

The overall project is made using Python 

IDE's Used-

- Jupyter Notebook 
- PyCharm 

In this project I have used Libraries and Modules like –

- Pandas, Numpy, Matplotlib
- Sklearn
 - Train_test_split
 - Logistic Regression
 - OneHot Encoder
 - Pipeline
 - Column Transformer

For this project I have used Streamlit Framework to make my web application

Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, NumPy, pandas, Matplotlib etc.

Data Set Used –

<https://www.kaggle.com/datasets/ramjidoolla/ipl-data-set>

Code Workflow

1) Importing necessary libraries

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

: match = pd.read_csv('matches.csv') #This project is a classification problem

: delivery = pd.read_csv('deliveries.csv')
```

2) Pre-processing Data

```
In [8]: #sorting how many runs were scored by a team in a match
delivery.groupby(['match_id','inning']).sum()['total_runs']

#here we can see that in 1st match - 1st team scored 207 runs; whereas
#2nd team scored 172 runs....similarly the same applies to all teams below in the list
```

```
Out[8]: match_id  inning
1             1      207
             2      172
2             1      184
             2      187
3             1      183
...
11413         2      170
11414         1      155
             2      162
11415         1      152
             2      157
Name: total_runs, Length: 1528, dtype: int64
```

```
In [9]: total_score_df = delivery.groupby(['match_id','inning']).sum()['total_runs'].reset_index()
total_score_df
```

750 rows x 3 columns

```
[11]: total_score_df=total_score_df[total_score_df['inning'] == 1 ]
```

```
[12]: match.merge(total_score_df[['match_id','total_runs']], left_on = 'id', right_on = 'match_id')
#merging the total runs scored by 1st team and match_id, in every match with the 'match' dataframe
```

```
[13]:
```

(Just added some main highlights here, you may refer to the Github code link at the end to see the full data pre- processing code)

```
In [13]: match_df = match.merge(total_score_df[['match_id','total_runs']], left_on = 'id', right_on = 'match_id')
```

Now there are some ipl teams which used to play before, but don't play now. So we will take only those teams which were common throughout the whole IPL, from starting. Also, we'll merge the details of the teams whose names have been changed, eg: Deccan Chargers is renamed to Sunrisers Hyderabad

```
In [14]: match_df['team1'].unique()
```

```
Out[14]: 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)
```

```
In [15]: teams = ['Sunrisers Hyderabad', 'Mumbai Indians',  
                 'Royal Challengers Bangalore',  
                 'Kolkata Knight Riders', 'Kings XI Punjab',  
                 'Chennai Super Kings', 'Rajasthan Royals',  
                 'Delhi Capitals']  
  
]
```

here we are replacing the old team names with the new ones

```
In [16]: match_df['team1'] = match_df['team1'].str.replace('Delhi Daredevils','Delhi Capitals')  
match_df['team2'] = match_df['team2'].str.replace('Delhi Daredevils','Delhi Capitals')
```

```
In [17]: match_df['team1'] = match_df['team1'].str.replace('Deccan Chargers','Sunrisers Hyderabad')  
match_df['team2'] = match_df['team2'].str.replace('Deccan Chargers','Sunrisers Hyderabad')
```

```
In [18]: match_df = match_df[match_df['team1'].isin(teams)]  
match_df = match_df[match_df['team2'].isin(teams)]  
#been the names/rows of only those teams whose names are mentioned
```

now we'll find out the number of wickets left after each ball

```
In [46]: delivery_df['player_dismissed'] = delivery_df['player_dismissed'].fillna("0") #convert the values to string of 0 where value = No  
delivery_df['player_dismissed'] = delivery_df['player_dismissed'].apply(lambda x:x if x == "0" else "1") #if the value is string  
delivery_df['player_dismissed'] = delivery_df['player_dismissed'].astype('int') #converting the whole column to integer
```

```
In [47]: delivery_df
```

3) Training and Testing our Model

```
In [63]: final_df.dropna(inplace=True)
```

```
In [64]: final_df = final_df[final_df['balls_left'] != 0]
```

```
In [65]: 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)
```

```
In [66]: X_train
```

Here we cleaned the data and removed null values, also, here we used **train_test_split** to train and test our model where **80% is sent to train and 20% data is sent to test**

4) Performing Logistic Regression on our given data and making a pipeline

Here we have created a pipeline and have used OneHotEncoder to categorize our data into 1s and 0s. Then we passed our data through Logistic Regression to predict the win probabilities

```
In [67]: 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')
```

```
In [68]: from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
```

```
In [69]: pipe = Pipeline(steps=[
    ('step1', trf),
    ('step2', LogisticRegression(solver = 'liblinear'))
])
```

```
In [70]: pipe.fit(X_train, y_train)
```

```
Out[70]: Pipeline(steps=[('step1',
                           ColumnTransformer(remainder='passthrough',
                                                transformers=[('trf',
                                                                OneHotEncoder(drop='first',
                                                                    sparse=False),
                                                                ['batting_team',
                                                                 'bowling_team', 'city'])])),
                           ('step2', LogisticRegression(solver='liblinear'))])
```

```
In [71]: y_pred = pipe.predict(X_test)
```

```
In [72]: from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
Out[72]: 0.8001261475926834
```

```
In [73]: pipe.predict_proba(X_test)[65]
```

```
Out[73]: array([0.61701498, 0.38298502])
```

5) Creating functions and Plotting Graphs

```
]: def match_progression(x_df,match_id,pipe):
    match = x_df[x_df['match_id'] == match_id] #extracting match_id
    match = match[(match['ball'] == 6)] #defining number of balls in a match
    new_df = match[['batting_team','bowling_team','city','runs_left','balls_left','wickets_left','actual_target_score','crr','rrr']]

    t1 = new_df['batting_team'].unique() #print the names of batting teams and bowling teams
    t2 = new_df['bowling_team'].unique()
    print(t1 + ' + ' + 'vs' + ' + ' + t2)

    new_df = new_df[new_df['balls_left'] != 0]
    result = pipe.predict_proba(new_df) #

    new_df['lose'] = np.round(result.T[0]*100,1) #finding lose probability after each over
    new_df['win'] = np.round(result.T[1]*100,1) #finding win probability after each over
    new_df['end_of_over'] = range(1,new_df.shape[0]+1) #completed over number

    target = new_df['actual_target_score'].values[0]
    runs = list(new_df['runs_left'].values)
    new_runs = runs[:]
    runs.insert(0,target)
    new_df['runs_after_over'] = np.array(runs)[:1] - np.array(new_runs) #runs after each over

    wickets = list(new_df['wickets_left'].values)
    new_wickets = wickets[:]
    new_wickets.insert(0,10)
    wickets.append(0)
    w = np.array(wickets)
    nw = np.array(new_wickets)
    new_df['wickets_in_over'] = (nw - w)[0:new_df.shape[0]] #wickets after each over

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

PLOTTING GRAPHS AND DETERMINING OVER BY OVER WIN/LOSE PROBABILITY FOR SOME EXAMPLE MATCH ID'S

Over By Over Data

```
new_df,target = match_progression(delivery_df,74,pipe)
new_df #data for match with match_id = 74
```

```
['Royal Challengers Bangalore vs Chennai Super Kings']
Target- 179
```

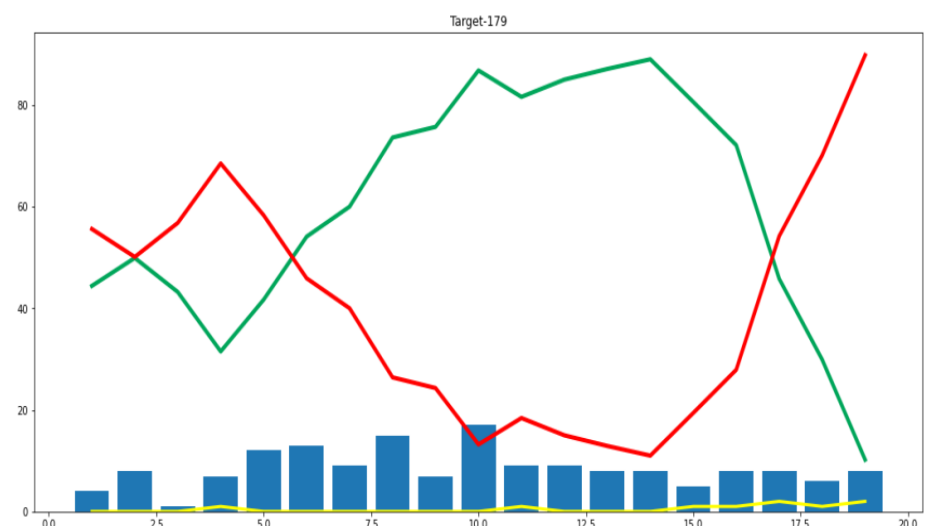
	end_of_over	runs_after_over	wickets_in_over	lose	win
10459	1	4	0	55.6	44.4
10467	2	8	0	50.1	49.9
10473	3	1	0	56.8	43.2
10479	4	7	1	68.5	31.5
10485	5	12	0	58.3	41.7
10491	6	13	0	45.9	54.1
10497	7	9	0	40.0	60.0
10505	8	15	0	26.4	73.6
10511	9	7	0	24.3	75.7
10518	10	17	0	13.2	86.8
10524	11	9	1	18.4	81.6
10530	12	9	0	15.0	85.0
10536	13	8	0	12.9	87.1
10542	14	8	0	11.0	89.0
10548	15	5	1	19.4	80.6
10555	16	8	1	27.9	72.1
10561	17	8	2	54.2	45.8
10567	18	6	1	70.1	29.9
10573	19	8	2	89.8	10.2

```
#Plotting graph for match with match_id = 74
```

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

```
Text(0.5, 1.0, 'Target-179')
```

For Match ID - 74



Over By Over Data

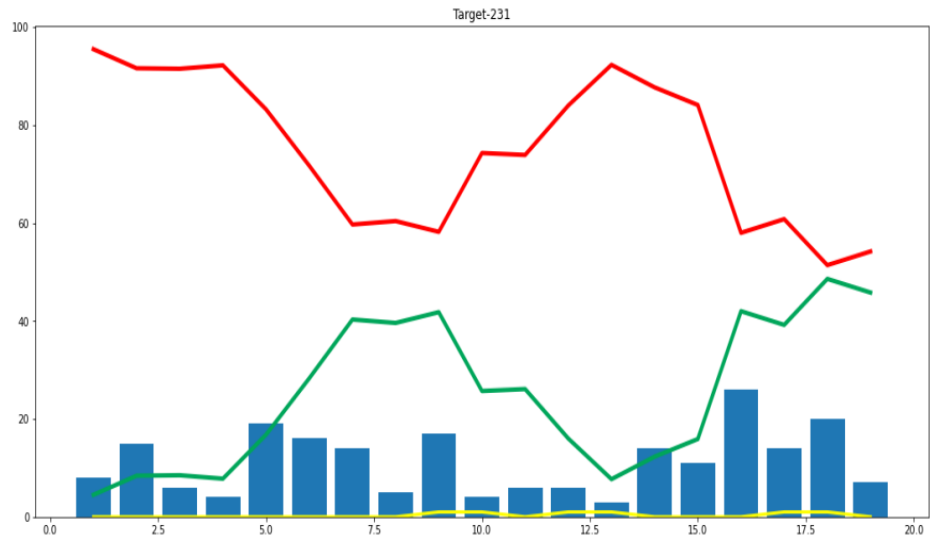
```
new_df,target = match_progression(delivery_df,50,pipe)
new_df #data for match with match_id = 50
```

['Mumbai Indians vs Kings XI Punjab']
Target- 231

	end_of_over	runs_after_over	wickets_in_over	lose	win
6215	1	8	0	95.5	4.5
6222	2	15	0	91.6	8.4
6228	3	6	0	91.5	8.5
6234	4	4	0	92.2	7.8
6241	5	19	0	83.2	16.8
6247	6	16	0	71.7	28.3
6253	7	14	0	59.7	40.3
6259	8	5	0	60.4	39.6
6265	9	17	1	58.2	41.8
6271	10	4	1	74.3	25.7
6277	11	6	0	73.9	26.1
6284	12	6	1	84.0	16.0
6290	13	3	1	92.3	7.7
6296	14	14	0	87.7	12.3
6302	15	11	0	84.1	15.9
6308	16	26	0	58.0	42.0
6315	17	14	1	60.8	39.2
6322	18	20	1	51.4	48.6
6328	19	7	0	54.2	45.8

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

: Text(0.5, 1.0, 'Target-231')



For Match ID - 50

Over By Over Data

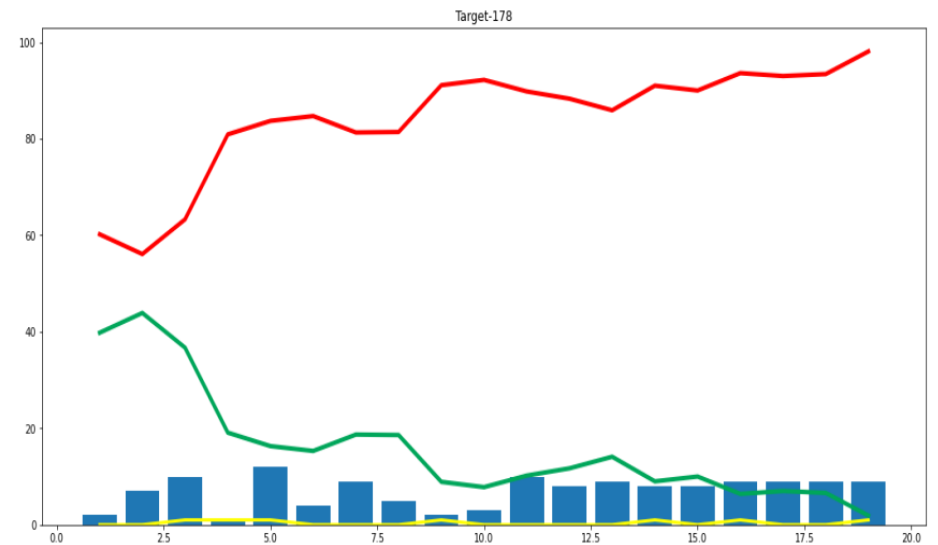
```
new_df,target = match_progression(delivery_df,200,pipe)
new_df #data for match with match_id = 200
```

['Kolkata Knight Riders vs Delhi Daredevils']
Target- 178

	end_of_over	runs_after_over	wickets_in_over	lose	win
39526	1	2	0	60.2	39.8
39533	2	7	0	56.1	43.9
39540	3	10	1	63.3	36.7
39546	4	1	1	80.9	19.1
39552	5	12	1	83.7	16.3
39558	6	4	0	84.7	15.3
39564	7	9	0	81.3	18.7
39570	8	5	0	81.4	18.6
39576	9	2	1	91.1	8.9
39582	10	3	0	92.2	7.8
39589	11	10	0	89.8	10.2
39596	12	8	0	88.3	11.7
39602	13	9	0	85.9	14.1
39608	14	8	1	91.0	9.0
39615	15	8	0	90.0	10.0
39621	16	9	1	93.6	6.4
39628	17	9	0	93.0	7.0
39634	18	9	0	93.4	6.6
39640	19	9	1	98.1	1.9

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

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



For Match ID - 200

Over By Over Data

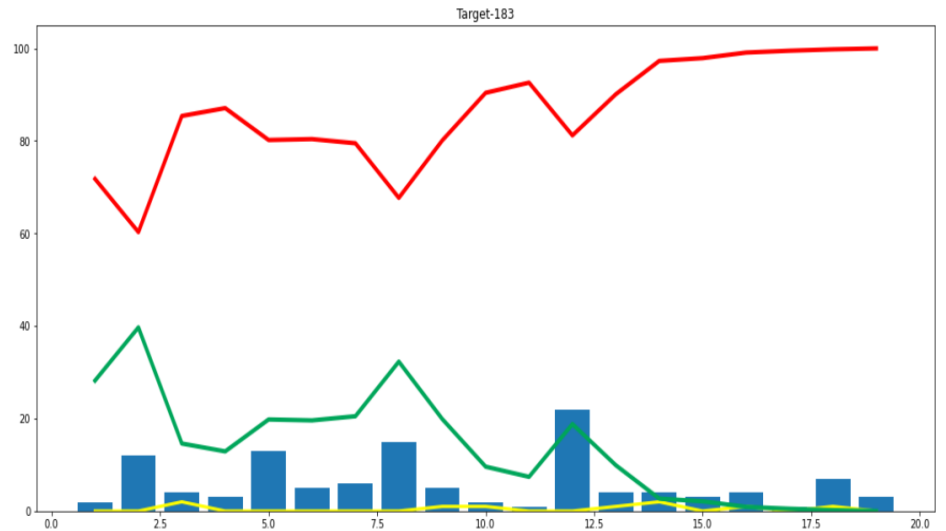
```
new_df,target = match_progression(delivery_df,69,pipe)
new_df #data for match with match_id = 69

['Mumbai Indians vs Kings XI Punjab']
Target- 183
```

	end_of_over	runs_after_over	wickets_in_over	lose	win
9298	1	2	0	71.8	28.2
9304	2	12	0	60.3	39.7
9311	3	4	2	85.4	14.6
9317	4	3	0	87.1	12.9
9324	5	13	0	80.2	19.8
9330	6	5	0	80.4	19.6
9336	7	6	0	79.5	20.5
9343	8	15	0	67.7	32.3
9350	9	5	1	80.1	19.9
9356	10	2	1	90.4	9.6
9362	11	1	0	92.6	7.4
9368	12	22	0	81.2	18.8
9374	13	4	1	90.1	9.9
9380	14	4	2	97.3	2.7
9386	15	3	0	97.9	2.1
9392	16	4	1	99.1	0.9
9398	17	1	0	99.5	0.5
9404	18	7	1	99.8	0.2
9411	19	3	0	100.0	0.0

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

Text(0.5, 1.0, 'Target-183')
```



For Match ID - 69

Over By Over Data

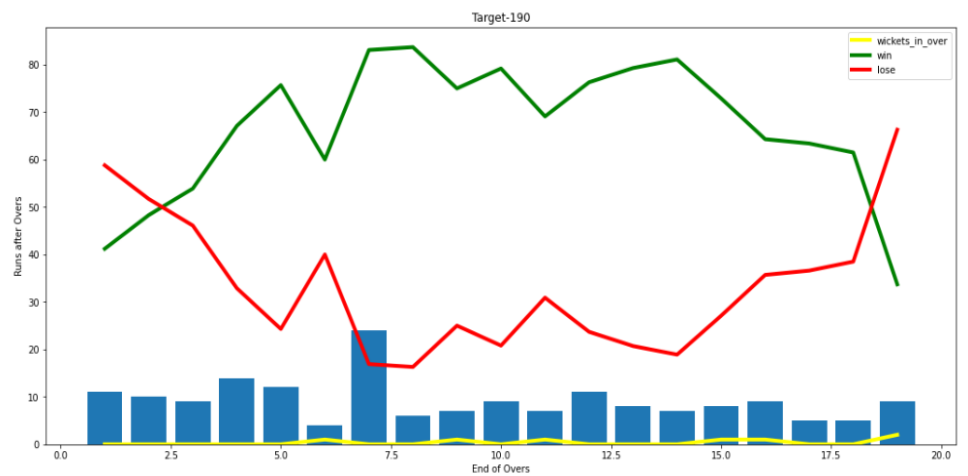
```
new_df,target = match_progression(delivery_df,125,pipe)
new_df #data for match with match_id = 125

['Chennai Super Kings vs Delhi Daredevils']
Target- 190
```

	end_of_over	runs_after_over	wickets_in_over	lose	win
21490	1	11	0	58.8	41.2
21496	2	10	0	51.7	48.3
21503	3	9	0	46.1	53.9
21509	4	14	0	32.9	67.1
21515	5	12	0	24.3	75.7
21521	6	4	1	40.0	60.0
21528	7	24	0	16.9	83.1
21535	8	6	0	16.3	83.7
21542	9	7	1	25.0	75.0
21548	10	9	0	20.8	79.2
21554	11	7	1	30.9	69.1
21560	12	11	0	23.7	76.3
21566	13	8	0	20.7	79.3
21572	14	7	0	18.9	81.1
21578	15	8	1	27.1	72.9
21585	16	9	1	35.7	64.3
21591	17	5	0	36.6	63.4
21597	18	5	0	38.5	61.5
21603	19	9	2	66.3	33.7

```
plt.figure(figsize=(18,8))
plt.plot(new_df['end_of_over'],new_df['wickets_in_over'],color='yellow',linewidth=4)
plt.plot(new_df['end_of_over'],new_df['win'],color='green',linewidth=4)
plt.plot(new_df['end_of_over'],new_df['lose'],color='red',linewidth=4)
plt.bar(new_df['end_of_over'],new_df['runs_after_over'])
plt.title('Target-' + str(target))
plt.legend(['wickets_in_over','win','lose'])
plt.xlabel('End of Overs')
plt.ylabel('Runs after Overs')
```

Text(0, 0.5, 'Runs after Overs')



For Match ID- 125

Overall Application Summary

In this project I have demonstrated the winning probability of a particular IPL team in a particular match situation, against an other team. Let's understand this with an example:

Suppose I want to predict the winning probability of a team named 'Sunrisers Hyderabad' (batting team). For that I will be entering the opponent team's (bowling team's) name, and batting team's details - host city, target score, current score, overs completed and fallen wickets;

And predict the winning probability for the batting team.

If the % of batting team is high, that means there are more chances for that team to win, else bowling team would win.

Cricket Win Probability Predictor - IPL

In this Project, we will be predicting the win probabilities of IPL Teams, where the input values are given by user for the team batting (i.e 2nd innings)

Select a Batting Team

Sunrisers Hyderabad

Select a Bowling Team

Mumbai Indians

Select Host City

Kolkata

Target Score

151.00

-

+

Current Score

49.00

-

+

Overs Completed

9.00

-

+

Wickets Fallen

1.00

-

+

Show Win Probabilities

Sunrisers Hyderabad - 54%

Mumbai Indians - 46%

Here we can see that given the values at a particular match point, Sunrisers Hyderabad has win probability of 54% whereas Mumbai Indians have win probability of 46%

Conclusion:

Through this project I learnt and explored about Classification problem, that is – Logistic Regression. I learnt how we can use logistic regression to predict the win probability of a cricket team give the input values from a particular match point. During this process I also enhanced my Data Pre-processing methodology.

Also I learnt about a new framework, that is, streamlit framework. This framework is similar to flask or django but more easier to use, as instead of reloading data everytime , it stores data in cache memory and loads it.

Finally, I would like to thank Jasmin Ma'am and Sphinx Worldbiz to provide me with this great internship opportunity, where I learnt a about Machine Learning during past weeks.

Project Code and Project Video Links:

Github : <https://github.com/RajK19/Sphinx-Worldbiz-Internship/tree/main/Project%202>

Youtube: <https://www.youtube.com/watch?v=Ckp1X7AQJqk>

Thank You!