COIMBATORE INSITUTE OF TECHNOLOGY

20MSSL12 Machine Learning Laboratory
CAT-PROJECT

Title: IPL Match Prediction

Team: 07

Members:

- 71762131011 – Bubalan S
- 71762131014 – Deshik S
- 71762131041 – Sai Nikhil S

Problem Statement:

The Indian Premier League (IPL) is one of the most popular and competitive T20 cricket leagues globally. With its inception in 2008, the tournament has witnessed significant growth, attracting top players and a massive fan base. The unpredictability of match outcomes makes it an interesting domain for predictive modelling.

Objective:

Develop a machine learning model to predict the outcome (win or loss) of IPL matches based on historical data from the years 2008 to 2019. The dataset includes information such as team performance, venue details, batting and bowling statistics, and other relevant features.

Approach:

1.Data Collection:

- Gather comprehensive historical data spanning from the inaugural season in 2008 to the latest available season.

- Include key parameters such as team performance, player statistics, venue details, and match outcomes.

2.Data Preprocessing:

- Cleanse and preprocess the data to handle missing values, outliers, and inconsistencies.

- Feature engineering to extract relevant information and create meaningful predictors.

3.Exploratory Data Analysis (EDA):

- Conduct in-depth EDA to uncover patterns, trends, and dependencies in the dataset.

- Visualize team performance, player contributions, and other relevant factors using plots and graphs.

4.Feature Selection:

- Identify and select the most influential features that contribute to match outcomes.

- Consider team strengths, player form, historical performance, and venue impact.

5.Model Selection:

- Choose appropriate machine learning models for prediction, considering the classification nature of match outcomes (win/loss).

- Evaluate models such as logistic regression, decision trees, random forests, or ensemble methods.

6.Model Training and Validation:

- Split the dataset into training and validation sets.

- Train the selected models using the training set and validate their performance on the validation set.

7. Hyperparameter Tuning:

- Fine-tune model hyperparameters to enhance predictive accuracy.

- Utilize techniques like grid search or randomized search for optimal parameter selection.

8. Evaluation Metrics:

- Evaluate model performance using metrics such as accuracy, precision, recall, and F1-score.

- Consider the business context and choose metrics that align with the objectives of predicting match outcomes.

9.Web Application Development:

- Implement a user-friendly web application using Streamlit for users to input match details and receive predictions.

- Incorporate visualizations and insights for a more engaging user experience.

10.Deployment and Monitoring:

- Deploy the trained model and web application for real-world use.

- Monitor model performance over time and update the dataset for continuous improvement.

Dataset:

The dataset comprises records of IPL matches from 2008 to 2019, including details like team names, player statistics, match venues, and outcomes.

- *Deleivers.csv*
- *Matches.csv*

Kaggle: *https://www.kaggle.com/datasets/ramjidoolla/ipl-data-set/data*

Attributes:

Project Overview:

Code Explanation

1. #importing Library
2. #importing csv file matches
3. #importing csv file delivery
4. #data sets contain only from 2008-2019 results
5. #contains each in detailed informationa of the bowling overs
   delivery.head()
6. #grouping the 2 datas
   total_score_df =
   delivery.groupby(['match_id','inning']).sum()['total_runs'].reset_index()
7. #combining the 1st innings of matchs data with total_score_df with common match id
   match_df =
   match.merge(total_score_df[['match_id','total_runs']],left_on = 'id' , right_on = 'match_id')
8. #Team 1 details name
   match_df['team1'].unique()
9. #only the current teams are taken

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

10. match_df['team1'] = match_df['team1'].str.replace('Delhi Daredevils','Delhi Capitals') – Change of the team name in data set
11. match_df['team1'] = match_df['team1'].str.replace('Deccan Chargers','Sunrisers Hyderabad') – change in the team name in both teams data set
12. #only the teams listed will be considered

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

13. # after considering only the 8 teams we get 641 from 756 match_df – displays the data after alter
14. # Duckworth-Lewis is based on the idea of compensating rain-affected teams for the loss of "run-scoring resources" #filtering the dl affected match_df= match_df[match_df['dl_applied'] == 0]
15. #consider the following tables from the dataset match_df = match_df[['match_id','city','winner','total_runs']]
16. #join the filtered data with deleviery csv delivery_df = match_df.merge(delivery,on='match_id')
17. #2nd innings delivery_df.shape
18. delivery_df.loc[:, 'total_runs_y'] = pd.to_numeric(delivery_df['total_runs_y'], errors='coerce')
19. unique_values = delivery_df['total_runs_y'].unique() print(unique_values) – This makes only the unique value from the selected row
20. #'delivery_df.groupby('match_id').cumsum()['total_runs_y']'
21. # Assuming 'total_runs_y' has been converted to numeric as discussed earlier delivery_df['total_runs_y'] = pd.to_numeric(delivery_df['total_runs_y'], errors='coerce') # Create the 'current_score' column delivery_df['current_score'] = delivery_df.groupby('match_id')['total_runs_y'].cumsum()
22. delivery_df – display the dataset
23. #to find the runs left tot of x - current score

```
    delivery_df['runs_left'] = delivery_df['total_runs_x'] -
    delivery_df['current_score']
24.#formula that is used for calculating the runs left with 1 run + 1st
    innings score
    delivery_df['balls_left'] = 126 - (delivery_df['over']*6 +
    delivery_df['ball'])
25.#wickets left code
    #making the nan to 0 in player dismissed
    delivery_df['player_dismissed'] =
    delivery_df['player_dismissed'].fillna("0")
    #if player dismissed by the other team mates its reg. as 1
    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')
26.# Assuming 'player_dismissed' column exists
    # Create the 'wickets' column
    wickets = delivery_df.groupby(['match_id',
    'player_dismissed']).cumcount() + 1
27.# crr = runs/overs – Current run rate
    delivery_df['crr'] = (delivery_df['current_score']*6)/(120 -
    delivery_df['balls_left'])
28.#rrr = runsleft*6/ballsleft – Required run rate
    delivery_df['rrr'] =
    (delivery_df['runs_left']*6)/delivery_df['balls_left']
29.#creating a function for result
    def result(row):
        return 1 if row['batting_team'] == row['winner'] else 0
30.#new col for result in the code
    delivery_df['result'] = delivery_df.apply(result,axis=1)
31.#resultant data frame
    final_df =
    delivery_df[['batting_team','bowling_team','city','runs_left','balls_le
    ft','wickets','total_runs_x','crr','rrr','result']]
32.#suffled order of the results

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

33.#missing values found
    final_df.isnull().sum()
```

34. #drop the values then if missing
    final_df.dropna(inplace = True)
35. X = final_df.iloc[:,:-1] # all rows except last col
    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)

Here in this line we start to feed the data for train it

36. ColumnTransformer - different transformations are required for different subsets of features in your dataset.
    Pipeline -process of constructing a sequence of data transformations followed by the application of a machine learning model.
    OneHotEncoder-one-hot encoding is a common technique to represent each category as a binary vector.
    # Assuming your trf transformation is correct
    trf = ColumnTransformer([
        ('trf', OneHotEncoder(sparse=False, drop='first'), ['batting_team', 'bowling_team', 'city'])
    ], remainder='passthrough')
    # Define the logistic regression model
    logistic_model = LogisticRegression()
    # Create the pipeline
    pipe = Pipeline([
        ('preprocessor', trf),
        ('classifier', logistic_model)
    ])
    # Train the model
    pipe.fit(X_train, y_train)
    # Make predictions
    predictions = pipe.predict(X_test)
37. #training the dataset
    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')

38. #logistic reg starts after train
    #steps of pipe line has ackno , after certain errors
    y_pred = pipe.predict(X_test)
39. #Accuracy score
    from sklearn.metrics import accuracy_score
    accuracy_score(y_test,y_pred)
40. ***#we use the logistic regression itself because Random forest is not accuracy and it wont predict on both sides***
41. ***def match_progression(x_df, match_id, pipe):***
    This function, `match_progression`, analyzes the progression of an IPL match identified by the given `match_id` using a pre-trained machine learning pipeline (`pipe`). It selects relevant features for a specific over (balls = 6) in the match, predicts the probability of winning and losing, and calculates the runs and wickets after each over. The resulting DataFrame `A` includes columns for the end of the over, runs scored in the over, wickets taken in the over, and the predicted probabilities of winning and losing. The target variable represents the total runs to chase. The function returns this processed DataFrame and the target variable for further analysis.
42. *import matplotlib.pyplot as plt*
    plt.figure(figsize=(18, 8))
    plt.plot(A['end_of_over'], A['wickets_in_over'], color='yellow', linewidth=3, label='Wickets in Over')
    plt.plot(A['end_of_over'], A['win'], color='#00a65a', linewidth=4, label='Win')
    plt.plot(A['end_of_over'], A['lose'], color='red', linewidth=4, label='Lose')
    plt.bar(A['end_of_over'], A['runs_after_over'], label='Runs After Over')
    plt.title('Target - ' + str(target))
    plt.legend()
    plt.show()
    We plot the statics for the purpose that depicts the trained data and displays the real time output with the report

Categorial Variable:

1. *Batting Team*
2. *Bowling Team*
3. *City*

4. *Runs left*
5. *Wickets left*
6. *Total runs x (Target)*
7. *CRR (Current run rate)*
8. *RRR (Required run rate)*

Input:

1. *Batting Team*
2. *Bowling Team*
3. *Current Score*
4. *Wickets Out*
5. *Overs Completed*
6. *Target*
7. *Venue*

Streamlit:

Streamlit is used to create an interactive web application that allows users to input match details and receive predictions. Here's a breakdown of how Streamlit is utilized in this project.
This Streamlit application serves as an IPL Win Predictor based on a pre-trained machine learning model. Users can input the batting team, bowling team, host city, target score, current score, overs completed, and wickets lost. Upon clicking the "Predict Probability" button, the application calculates the probability of the selected batting team winning and the bowling team losing. The predictions are displayed with rounded percentages for user-friendly interpretation, providing cricket enthusiasts with a quick and interactive tool for predicting match outcomes.

Features:

1. Team information (batting team, bowling team)

2. Venue details (city, stadium)

3. Match statistics (runs left, balls left, wickets)

4. Inning-wise details (runs scored, wickets taken)

5. Historical performance metrics

Target Variable:

The target variable is binary:

1: The team batting first won the match.

0: The team batting second won the match.

Tasks:

1. Data Preprocessing:

- Handle missing values.

- Convert categorical variables into a suitable format.

- Explore and visualize data for insights.

2. Feature Engineering:

- Create relevant features that might impact match outcomes.

- Normalize or scale features as necessary.

3. Model Selection:

- Choose an appropriate classification algorithm [Logistic Regression, Random Forest (Case-Study)] for predicting match results.

4. Model Training:

- Train the selected model on historical data.

5. Model Evaluation:

- Evaluate the model's performance using appropriate metrics (accuracy, precision, recall, F1-score).

6. Prediction:

- Use the trained model to predict match outcomes for new or unseen data.

Evaluation Criteria:

The model will be evaluated based on its accuracy in predicting match outcomes. Additional metrics such as precision, recall, and F1-score can provide insights into the model's performance on specific classes.

Deliverables:

1. Jupyter notebook contains the code.

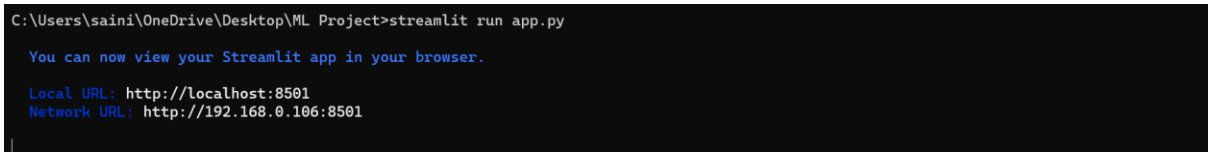
2. Documentation describing the data preprocessing, feature engineering, model selection, and evaluation steps.

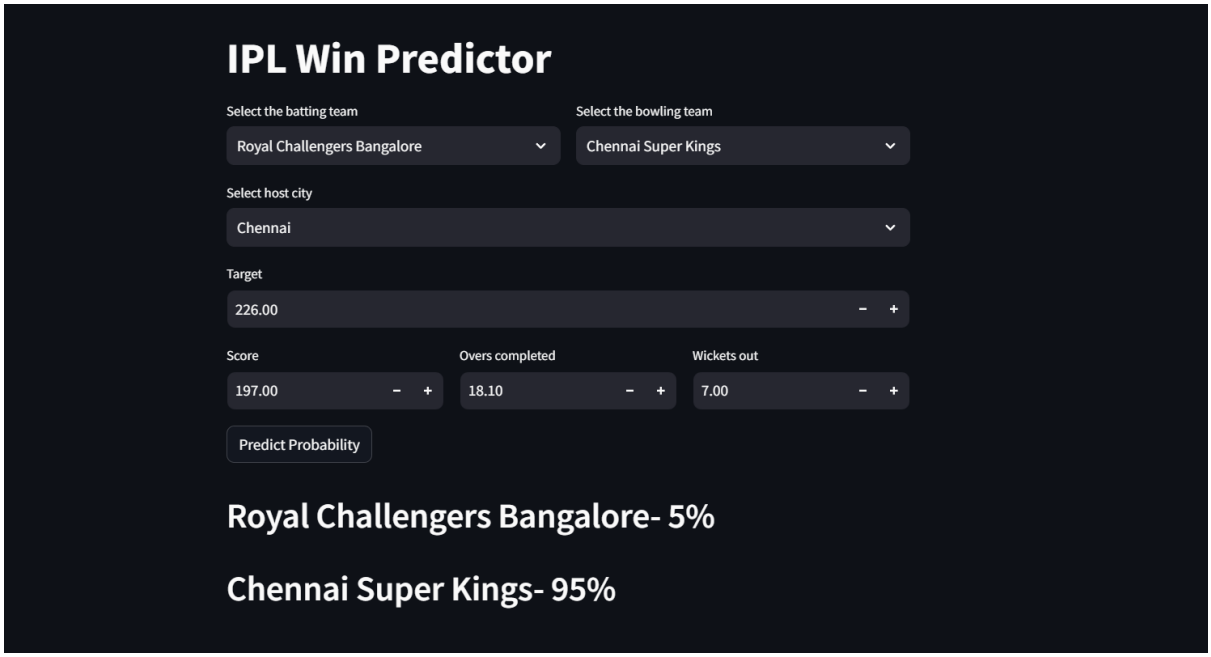
3. Visualizations and insights gained from the data exploration.

4. Trained machine learning model for predicting IPL match results.

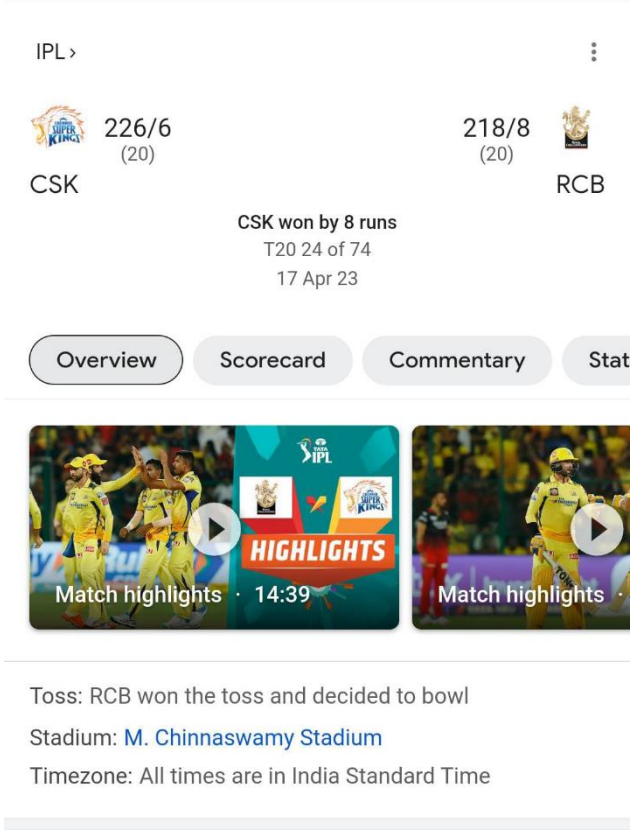
Outcome:

Run the App.py in streamlit





Real – World Outcome:



Fall of wickets

6/1 (V. Kohli, 0.4 ov) · 15/2 (M. Lomror, 1.6 ov) · 141/3 (G. Maxwell, 12.1 ov) · 159/4 (Faf du Plessis, 13.6 ov) · 191/5 (D. Karthik, 16.5 ov) · 192/6 (S. Ahmed, 17.1 ov) · 197/7 (W. Parnell, 18.1 ov) · 218/8 (S. Prabhudessai, 19.6 ov)

Filenames:

- Untitled.py – Main file for logistics
- Datasets – Deliveries.csv, matches.csv
- Streamlit – app.py

References:

Kaggle: https://www.kaggle.com/code/prashant111/logistic-regression-classifier-tutorial

Kaggle: https://www.kaggle.com/code/frankmollard/machine-learning-process-idea-2-app?scriptVersionId=151531667

Conclusion:

Data-Driven Solution: *Leveraged data science and machine learning techniques to build a user-friendly web app using Streamlit for predicting IPL match results.*

Predictive Model: *Developed a robust predictive model by analysing historical IPL data, offering quick and accurate forecasts based on crucial match features.*

User Accessibility: *Streamlit's intuitive interface ensures accessibility for cricket enthusiasts and analysts, promoting easy interaction with the predictive model.*

Sports Analytics Showcase: *Demonstrates the potential of machine learning in sports analytics, emphasizing the value of intuitive and interactive platforms for insights delivery.*

Transformative Impact: *As cricket fans actively engage with the app, it signifies the transformative impact of data-driven decision-making in shaping the landscape of sports analytics.*