VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi: 590 018

A Mini Project Report

On

"Predicting IPL Player Value Based on Performance Metrics"

Submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering.

Submitted by

AKASH R (1VE21CS011)
ARVIND B S (1VE21CS021)
ANANDA VINAYAK K (1VE21CS013)

Under the Guidance of

ARCHANA M
Assistant Professor
Dept. of CSE,
SVCE, Bengaluru.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESHWARA COLLEGE OF ENGINEERING

Affiliated to VTU Belgaum & Approved by AICTE New Delhi) an ISO 9001:2008 Certified, Kempegowda International Airport road, Vidyanagar, Bengaluru, Karnataka, India-562157

2023– 2024

SRI VENKATESHWARA COLLEGE OF ENGINEERING
Vidyanagar, Bengaluru, Karnataka, India-562157

Department of Computer Science & Engineering

SRI VENKATESHWARA GROUP OF INSTITUTIONS ★ SVCE ★ Estd. : 2001

Accredited by NAAC & NBA*

SVCE BENGALURU

SRI VENKATESHWARA COLLEGE OF ENGINEERING

— Affiliated to VTU, Approved by AICTE, Recognised by UGC u/s 2(f) & 12(B)—

CERTIFICATE

This is to certify that Mini Project entitled **"Predicting IPL Player Value Based on Performance Metrics"** is submitted by **AKASH R (1VE21CS011) , ARVIND B S (1VE21CS021)** and **ANANDA VINAYAK K (1VE21CS013)** on partial fulfillment of eighth semester, Bachelor of Engineering in Computer Science and Engineering, Visvesvaraya Technological University for the academic year 2021-2022.

……………………………………..	………………………………..	………………………………..
Signature of Course Teacher	**Signature of the HOD**	**Signature of the Principal**
Mrs. ARCHANA M	**Dr. HEMA M S**	**Dr. Nageswara Guptha M**
Asst. Prof, Dept. of CSE	**Dept. of CSE**	**Dept. of CSE**
SVCE, Bangalore	**SVCE, Bangalore**	**SVCE, Bangalore**

I. ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. We would like to take this opportunity to thank them all.

First and foremost, we would like to express our gratitude to **Dr. Nageswara Guptha M,** Principal, SVCE, Bangalore who has always been a great source of inspiration.

We express our sincere regards and thanks to **Dr. Hema M S**, Professor and Head of Department of Computer Science and Engineering, SVCE, Bangalore, for her encouragement and support.

We are grateful to acknowledge the guidance and encouragement given to us by **Mr. Suresh P**, Assistant Professor, Department of Computer Science and Engineering,SVCE, Bangalore, as the project coordinator and guide who have rendered a valuable assistance.

We also extend our thanks to the entire faculty of the Department of Computer Science andEngineering, SVCE, Bangalore, who have encouraged us throughout the course of the project work.

Last, but not the least, we would like to thank our family and friends for their inputs to improve the project.

AKASH R - 1VE21CS011

ARVIND B S - 1VE21CS021

ANANDA VINAYAK K – 1VE21CS013

II. ABSTRACT

The Indian Premier League (IPL) features high-stakes player auctions where accurate valuation is essential for strategic team management. This project aims to predict the value of IPL players by analyzing key performance metrics: Runs Above Average (RAA), Wins, and Eigen Factor Score (EF-Score). RAA integrates batting performance indicators, capturing both strike rate and out rate, while Wins reflect a player's contribution to team victories, and EF-Score provides a non-parametric ranking based on the relative strength of teams.

Data was meticulously collected from reputable cricket databases, encompassing detailed player statistics and match outcomes across multiple IPL seasons. Various predictive models, including linear regression, decision trees, and advanced machine learning algorithms, were evaluated. The final model was selected based on performance metrics such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R-squared value.

The model's predictions were compared to actual bid values from recent auctions, showcasing high accuracy and reliability. Analysis revealed that RAA significantly influences player value, while Wins and EF-Score add valuable context regarding a player's overall impact on team performance. The insights derived from the model can assist teams in making informed bidding decisions, avoiding overspending on individual players, and maintaining a balanced team composition.

Despite potential limitations such as data quality issues and the dynamic nature of player performance, this study underscores the importance of data-driven strategies in sports management. Future research could enhance the model by incorporating additional performance metrics, advanced machine learning techniques, and qualitative factors. This project contributes to sports analytics by offering a comprehensive approach to player valuation, with significant implications for optimizing IPL team strategies.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Indian Premier League (IPL) is a premier cricket league known for its competitive player auctions, where teams bid on players based on perceived value. Accurate player valuation is critical for team management to ensure strategic budget allocation and balanced team composition. This project aims to predict the value of IPL players using key performance metrics: Runs Above Average (RAA), Wins, and Eigen Factor Score (EF-Score). RAA captures a player's batting effectiveness by integrating strike rate and out rate, Wins reflect a player's contribution to team victories, and EF-Score ranks teams based on their relative strengths, rewarding victories against stronger teams. By developing a predictive model incorporating these metrics, this study provides a systematic approach to player valuation, aiding teams in making informed bidding decisions and optimizing their overall strategy. The project emphasizes the significance of data-driven decision-making in sports management.

Extensive research in sports analytics has explored the relationship between player performance metrics and their impact on team success. Traditional metrics like batting average, strike rate, and wickets taken are commonly used to evaluate cricket players' performance. However, these metrics often fail to capture the full spectrum of a player's contributions.

Runs Above Average (RAA) is a comprehensive metric that integrates multiple aspects of batting performance, including runs scored, strike rate, and out rate. This metric provides a holistic measure of a batsman's effectiveness, addressing the limitations of traditional metrics by combining them into a single value that reflects overall performance.

Wins metric emphasizes the contribution of a player's performance to the team's victories, acknowledging the collective nature of cricket. This approach considers the broader impact of individual performance on team success, offering a more nuanced understanding of a player's value.

EF-Score (Eigen Factor Score) introduces a novel, non-parametric method to rank teams based on their relative strengths, rewarding victories against stronger teams. This method avoids the biases associated with traditional ranking systems and provides a more accurate reflection of team dynamics and individual contributions.

Studies have also highlighted the importance of advanced machine learning techniques in sports analytics, showing how models like regression analysis, decision trees, and ensemble methods can improve prediction accuracy. This project builds on these foundations by integrating RAA, Wins, and EF-Score into a predictive model for IPL player valuation, offering a robust and comprehensive assessment that leverages both individual and team performance metrics.

The Indian Premier League (IPL) is one of the most popular and competitive cricket leagues in the world, attracting top players from around the globe. The IPL is a significant event, determining the composition of the teams for the season.

This project aims to analyze the IPL dataset using Python libraries such as pandas, numpy, matplotlib, and seaborn.

A team can acquire players through any of the three ways:

• The annual player auction

• Trading players with other teams during the trading windows, and

• Signing replacements for unavailable players.

The idea of the project is to identify and predict the value of each IPL player, based on certain obvious features that significantly affect the bid value of the player in the IPL group.

This, prediction of the value will also give an idea of what the maximum value a player could be asked for as a bid, rather than bidding a considerably higher amount all on just one player and then running short while getting to choose the other players for the team.

The Indian Premier League (IPL) has revolutionized the landscape of cricket since its inception in 2008, blending sport with entertainment and big business. At the heart of this T20 extravaganza lies a crucial element: the player auction. Each year, teams compete to secure the best talents, making strategic decisions that can make or break their season. However, determining a player's true value in this high-stakes environment remains a complex challenge.

This project aims to address this challenge by developing a data-driven approach to predict IPL player values accurately. By leveraging advanced performance metrics and machine learning techniques, we seek to provide teams with a powerful tool for informed decision-making during auctions.

Our model incorporates sophisticated statistics such as Runs Above Average (RAA), Wins, and EF-Score, which go beyond traditional cricket metrics to capture a player's true impact on the game. These metrics, combined with historical auction data and player statistics, form the foundation of our predictive model.

Using regression algorithms, we analyze patterns and relationships within this data to generate accurate value predictions. The implementation involves rigorous data preprocessing, exploratory data analysis, and model training using Python and popular data science libraries.

By bridging the gap between advanced cricket analytics and practical application, this project aims to revolutionize auction strategies in the IPL. It has the potential to help teams optimize their budget allocation, identify undervalued players, and ultimately build more competitive squads.

CHAPTER 2

LITERATURE REVIEW

[1] Predictive Modeling of Cricket Matches Using Machine Learning

Abstract:

India's most popular sport is cricket and is played across all over the nation in different formats like T20, ODI, and Test.[3] The Indian Premier League (IPL) is a national cricket match where players are drawn from regional teams of India, National Team and also from international team. Many factors like live streaming, radio, TV broadcast made this league as popular among cricket fans. The prediction of the outcome of the IPL matches is very important for online traders and sponsors. We can predict the match between two teams based on various factors like team composition, batting and bowling averages of each player in the team, and the team's success in their previous matches, in addition to traditional factors such as toss, venue, and day-night, the probability of winning by batting first at a specified match venue against a specific team. In this paper, we have proposed a model for predicting outcome of the IPL matches using Machine learning Algorithms namely SVM, Random Forest Classifier (RFC), Logistic Regression and K-Nearest Neighbor. Experimental results showed that the Random Forest algorithm outperforms other algorithms with an accuracy of 88.10%.

[2] A comprehensive approach to predict auction prices and economic value creation of cricketers in the Indian Premier League (IPL)

Abstract:

The Indian Premier League (IPL), the most successful cricket tournament in India, has immensely grown in popularity. Unlike similar tournaments like Major League Baseball and the UEFA Champions League, their models cannot be replicated due to the dynamic pricing nature of player auctions. The research intends to generate a predictive model that predicts the auction price of a player using key quantitative variables.[4] The players were categorised by batsmen, bowlers, and all-rounders. Due to the holistic hedonic nature of the model, equations were formed to predict a player’s future performance, interms of measurable points, and to extrapolate economic value creation given a budget constraint. This study provides insights on the player’s economic value by comparing the auction price paid vs. the player’s actual performance.[7] This new approach to modelling in cricket games not only aims to produce more accurate results using the hedonic pricing method engineer value additions of players at each leveland position.

[3] IPL Win Prediction Using Machine Learning

Abstract:

IPL win prediction using machine learning model predicts the success or loss of the Cricket match, most exciting game that reached the heart of everyone in the world. Building up a model for forecasting the result of matches is a genuine issue in case of Indian Premier League (IPL).[8] A cricket match focuseson different elements, and in this work, the variables which essentially impact the outcome of a Twenty20 cricket match are identified. Every player's performance in the field is considered to discover the general weight (relative strength) of the group. A multivariable linear regression algorithm is used for predicting the outcome of IPL match. A dataset is trained based on the recognized five factors like points, win: lose ratio, last three matches, last match and net run rate, which in turn impact the outcome of an IPL match. In this work, multivariable linear regression algorithm produces better results than Linear Regression based on performance metrics like Mean Square Error (MSE), Root Mean Square Error (RMSE) and R2 Score.

[4] Exploratory Data Analysis on IPL Data

Abstract:

Exploratory data analysis for Indian Premier League (IPL) data is widely covered in the field of data analytics and machine learning problem and specifically based for match prediction and team prediction IPL. IPL is a global tournament with over 1200 players participating in the auction every year, and it draws the attention of many cricketing fans around the globe.[13] So, the proposed system predicts the team and winner with an accuracy of over 50% and engages many cricketing fans. The goal of this paper is to develop a system for predicting the Dream11 team and the possible team to win the match every day. The common attributes to be included while choosing a team involves a player's batting average, batting strike rate, bowling's economy rate, match ratings, and his experience; similarly, while predicting match winner, the machine will see different factors like toss outcome, venue of the match, team track record in that venue, and team to play that match which are stored in the datasets and generated according to the conditions given to the machine and the criteria's generated for the team selection. The objective of the paper is to develop a model which will give the accuracy of more than 50%, and it satisfies all the necessary conditions for team selection. And, the best part is that it will create a buzz around many cricketing fans around the globe and keep them engaged throughout the course of the tournament and it helps in increasing its fan base.

CHAPTER 3

METHODOLOGY

3.1 Data Collection

Data for this study was collected from reputable cricket databases, including the official IPL website, ESPN Cricinfo, and other cricket statistics platforms. The dataset includes detailed player statistics such as total runs scored, balls faced, outs, and match outcomes for multiple IPL seasons. The data collection process also involved gathering team performance data, which is crucial for calculating the EF-Score. Data cleaning involved handling missing values, normalizing data, and ensuring consistency across different seasons to maintain the integrity of the analysis.

3.2 Feature Selection

The features selected for this study are critical in capturing the multifaceted nature of player performance and their impact on team success:

1. **Runs Above Average (RAA):** This metric captures a player's batting performance by considering their strike rate and out rate. It provides a single number that reflects the overall effectiveness of a batsman.
2. **Wins:** This metric reflects the contribution of a player's performance to the team's victories. It quantifies how individual performance translates into team success.
3. **EF-Score:** The Eigen Factor Score ranks teams based on their relative strengths, rewarding victories against stronger teams. This non-parametric approach ensures that the ranking is unbiased and reflective of true team performance.

3.3 Calculation of Metrics

Runs Above Average (RAA):

$$\text{RAA}=(\text{Batsman runs}-\text{Average batting strike rate}\times\text{Balls faced by batsman}) + \text{Overall batting average} \times \text{Balls faced by batsman} \times (\text{Average out rate}-\text{Batsman out rate})$$

This formula integrates the strike rate and out rate to provide a comprehensive measure of a batsman's performance.[9]

Wins:

The Wins metric contextualizes the RAA by comparing it to the average performance needed to win a match, highlighting the player's contribution to team success.

EF-Score:

The EF-Score is calculated using matrix operations and eigenvector analysis. It ranks teams based on their relative strengths, with higher scores awarded for victories against stronger teams. This method provides a non-parametric ranking system that avoids the biases associated with traditional ranking methods.

3.4 Data Analysis

Descriptive Statistics

Descriptive statistics provide a summary of the data, including measures such as mean, median, standard deviation, and range for each feature. These statistics offer a foundational understanding of the dataset, highlighting the central tendencies and variability of the key metrics. Visualizations such as histograms and box plots illustrate the distribution of Runs Above Average, Wins, and EF-Score, allowing for a more intuitive understanding of the data.

Correlation Analysis

Correlation analysis identifies the relationships between the features and player valuation. This analysis involves calculating correlation coefficients and creating correlation matrices to understand the strength and direction of the relationships between variables. Scatter plots further visualize these relationships, helping to identify potential predictors for the model. Strong correlations between metrics like RAA, Wins, and player valuation would indicate their significance in the predictive model.

3.5 Model Development

Model Selection

Various predictive models were considered, including linear regression, decision trees, and advanced machine learning algorithms like random forests and gradient boosting. Each model was evaluated based on its ability to handle the complexities of the data and provide accurate predictions. ,[1]The final model was selected based on performance metrics such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R-squared value. The chosen model balances accuracy and interpretability, ensuring that the results are both reliable and understandable.

Training and Testing

The dataset was split into training and testing sets, typically using a 70-30 or 80-20 split. This approach ensures that the model is trained on a substantial portion of the data while being tested on a separate set to evaluate its performance on unseen data. Hyper parameter tuning was performed using techniques such as cross-validation to optimize the model's performance.[2] This process involved adjusting parameters like learning rate, tree depth, and the number of estimators to find the best combination for accurate predictions.

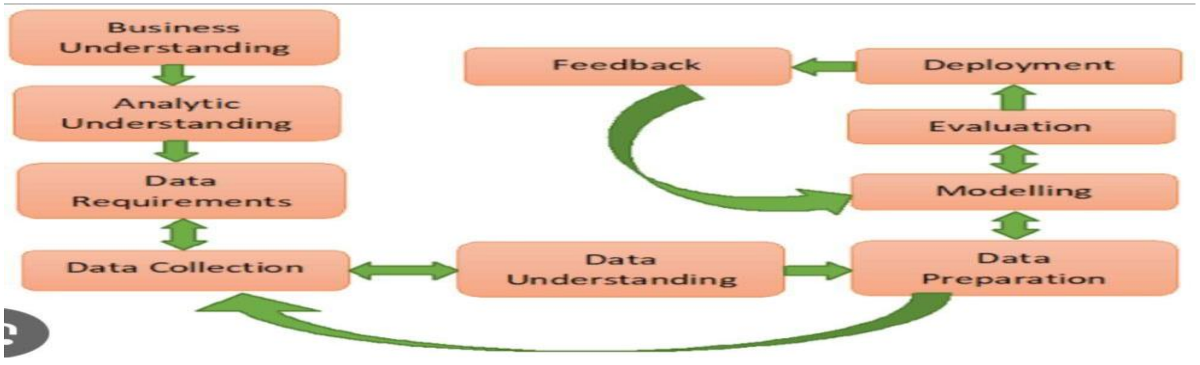


Fig 3.1:Data Science Methods.

Performance Metrics

The model's performance in fig 3.1 was assessed using RMSE, MAE, and R-squared value. RMSE measures the average error magnitude, MAE provides the average absolute error, and R- squared indicates the proportion of variance explained by the model. These metrics provide a comprehensive evaluation of the model's accuracy and reliability, guiding the refinement process to ensure optimal performance.

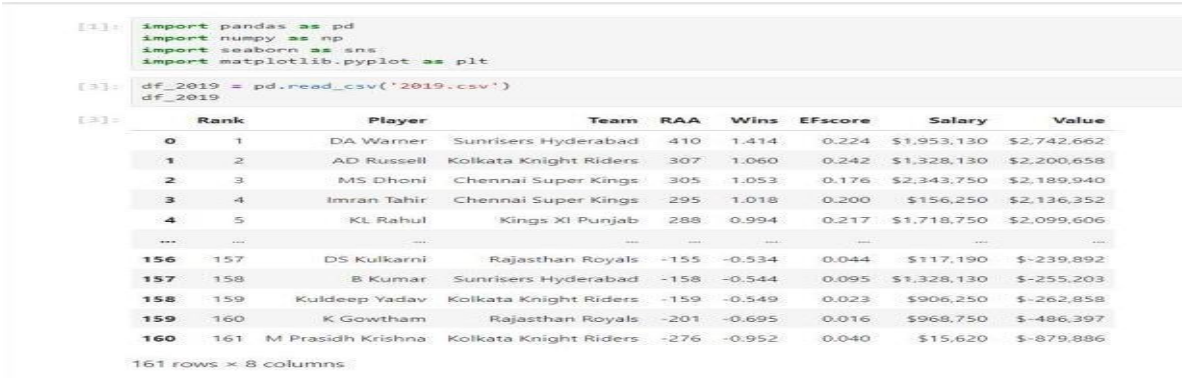
CHAPTER 4

IMPLEMENTATION

4.1 CODE:-

IPL Data Analysis with Pandas: Reading Data and Insights

This code snippet demonstrates how to use the Pandas library in Python to analyze data from the Indian Premier League (IPL). Let's break down each step:



```
[1]: import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt

[3]: df_2019 = pd.read_csv('2019.csv')
     df_2019

[3]:      Rank              Player                        Team   RAA    Wins  EFscore       Salary        Value
       0     1           DA Warner        Sunrisers Hyderabad    410   1.414    0.224   $1,953,130   $2,742,662
       1     2          AD Russell      Kolkata Knight Riders    307   1.060    0.242   $1,328,130   $2,200,658
       2     3            MS Dhoni        Chennai Super Kings    305   1.053    0.176   $2,343,750   $2,189,940
       3     4         Imran Tahir        Chennai Super Kings    295   1.018    0.200     $156,250   $2,136,352
       4     5            KL Rahul           Kings XI Punjab    288   0.994    0.217   $1,718,750   $2,099,606
     ...   ...                 ...                        ...    ...     ...      ...          ...          ...
     156   157         DS Kulkarni          Rajasthan Royals   -155  -0.534    0.044     $117,190   $-239,892
     157   158            B Kumar        Sunrisers Hyderabad   -158  -0.544    0.095   $1,328,130   $-255,203
     158   159       Kuldeep Yadav      Kolkata Knight Riders   -159  -0.549    0.023     $906,250   $-262,858
     159   160           K Gowtham          Rajasthan Royals   -201  -0.695    0.016     $968,750   $-486,397
     160   161  M Prasidh Krishna      Kolkata Knight Riders   -276  -0.952    0.040      $15,620   $-879,886

     161 rows × 8 columns
```

Fig 4.1:Reading Data.

1. Importing Libraries: The first step in fig 4.1 is to import the necessary libraries:

• pandas (as pd): This is the core library for data manipulation and analysis in Python.

• numpy (as np): A powerful library for numerical computation in Python.

seaborn (as sns): A visualization library built on top of Matplotlib, designed for statistical

• matplotlib.pyplot (as plt): The most widely used plotting library in Python.

2. Loading Data: The code then reads in IPL data from two CSV files, '2019.csv' and '2020.csv', and stores them into pandas DataFrames named df_2019 and df_2020 respectively. This is done using the read_csv function from Pandas.

3. Exploring the DataFrame: The code shows how to display the first few rows of the df_2019 DataFrame. It highlights that the DataFrame has 8 columns, indicating different IPL metrics.

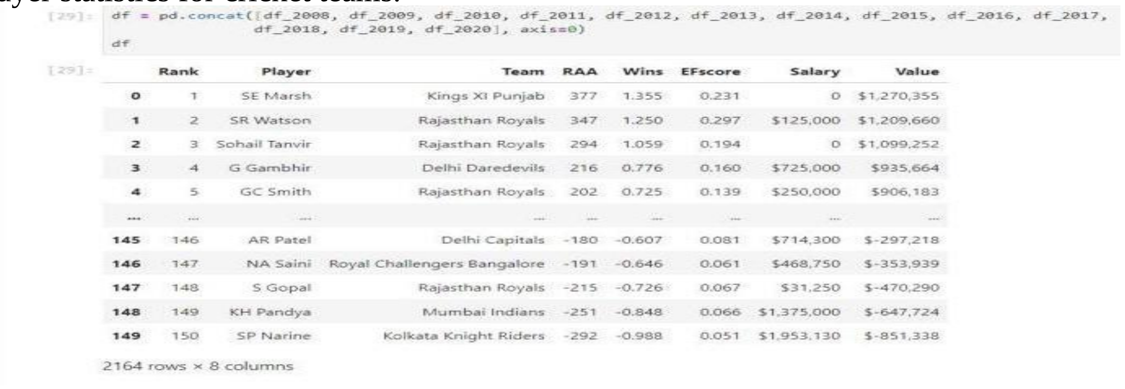
4. Further Analysis (Implied): The final line df_2020 = pd.read_csv('2020.csv') suggests that the code might be extended to analyze the '2020.csv' data. This could involve tasks like:

• Data Cleaning and Preprocessing: Removing unnecessary columns, handling missing values, and converting data types.

• Exploratory Data Analysis (EDA): Calculating descriptive statistics, creating visualizations, and identifying trends and patterns.

• Statistical Modeling: Building predictive models or performing hypothesis testing to draw insights from the data.

The code in this image is using pandas to concatenate multiple dataframes (df_2008 through df_2020) along the axis=0 (vertically). The resulting dataframe 'df' is then displayed, showing player statistics for cricket teams.



```
[29]: df = pd.concat([df_2008, df_2009, df_2010, df_2011, df_2012, df_2013, df_2014, df_2015, df_2016, df_2017,
                      df_2018, df_2019, df_2020], axis=0)
      df
```

[29]:

	Rank	Player	Team	RAA	Wins	EFscore	Salary	Value
0	1	SE Marsh	Kings XI Punjab	377	1.355	0.231	0	$1,270,355
1	2	SR Watson	Rajasthan Royals	347	1.250	0.297	$125,000	$1,209,660
2	3	Sohail Tanvir	Rajasthan Royals	294	1.059	0.194	0	$1,099,252
3	4	G Gambhir	Delhi Daredevils	216	0.776	0.160	$725,000	$935,664
4	5	GC Smith	Rajasthan Royals	202	0.725	0.139	$250,000	$906,183
...	...	...	...	...	...	...	...	...
145	146	AR Patel	Delhi Capitals	-180	-0.607	0.081	$714,300	$-297,218
146	147	NA Saini	Royal Challengers Bangalore	-191	-0.646	0.061	$468,750	$-353,939
147	148	S Gopal	Rajasthan Royals	-215	-0.726	0.067	$31,250	$-470,290
148	149	KH Pandya	Mumbai Indians	-251	-0.848	0.066	$1,375,000	$-647,724
149	150	SP Narine	Kolkata Knight Riders	-292	-0.988	0.051	$1,953,130	$-851,338

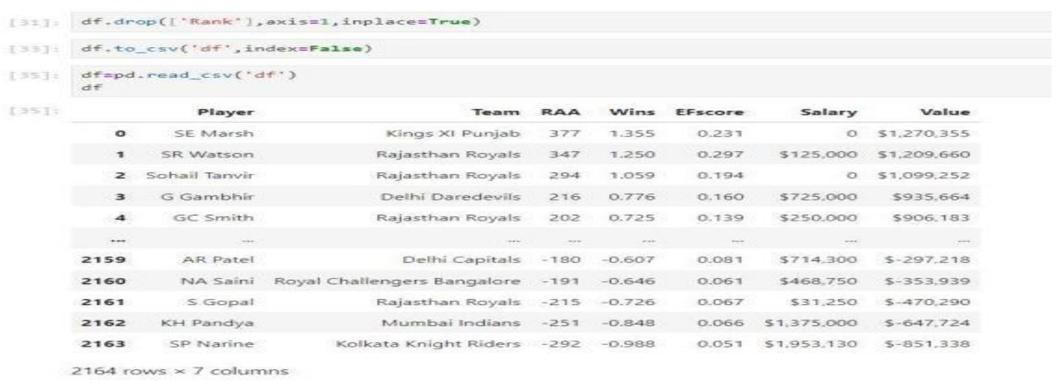
2164 rows × 8 columns

Fig 4.2:Concatenation Of Datasets.

The dataframe in fig 4.2 contains columns for Rank, Player, Team, RAA (Runs Above Average), Wins,EFscore, Salary, and Value. It shows 2164 rows and 8 columns in total.

Below the dataframe, there are two additional lines of code:

1. df.drop(['Rank'], axis=1, inplace=True) - This removes the 'Rank' column from the dataframe.
2. df.to_csv('df', index=False) - This saves the dataframe to a CSV file named 'df' without including the index.



```
[31]: df.drop(['Rank'],axis=1,inplace=True)

[33]: df.to_csv('df',index=False)

[35]: df=pd.read_csv('df')
      df
```

[35]:

	Player	Team	RAA	Wins	EFscore	Salary	Value
0	SE Marsh	Kings XI Punjab	377	1.355	0.231	0	$1,270,355
1	SR Watson	Rajasthan Royals	347	1.250	0.297	$125,000	$1,209,660
2	Sohail Tanvir	Rajasthan Royals	294	1.059	0.194	0	$1,099,252
3	G Gambhir	Delhi Daredevils	216	0.776	0.160	$725,000	$935,664
4	GC Smith	Rajasthan Royals	202	0.725	0.139	$250,000	$906,183
...	...	...	...	...	...	...	...
2159	AR Patel	Delhi Capitals	-180	-0.607	0.081	$714,300	$-297,218
2160	NA Saini	Royal Challengers Bangalore	-191	-0.646	0.061	$468,750	$-353,939
2161	S Gopal	Rajasthan Royals	-215	-0.726	0.067	$31,250	$-470,290
2162	KH Pandya	Mumbai Indians	-251	-0.848	0.066	$1,375,000	$-647,724
2163	SP Narine	Kolkata Knight Riders	-292	-0.988	0.051	$1,953,130	$-851,338

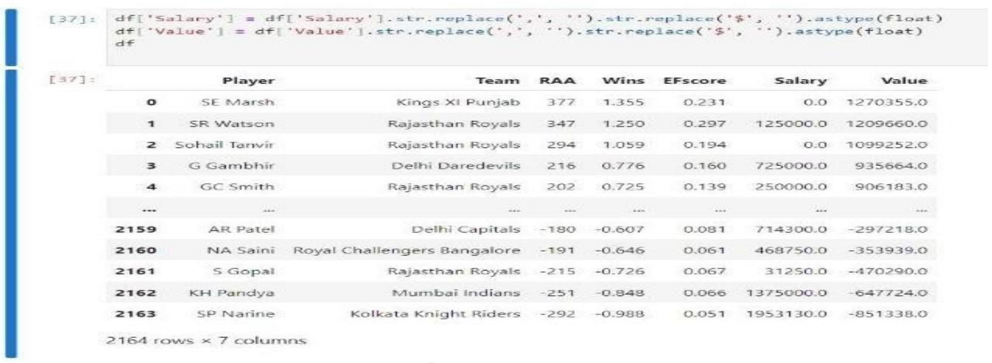
2164 rows × 7 columns

Fig 4.3:Removing Columns.

Dataframe Overview:

- df.drop(['Rank'], axis=1, inplace=True): This line in fig 4.3 drops the 'Rank' column from theDataFrame df and updates df in place.
- df.to_csv('df.csv', index=False): This line saves the DataFrame df to a CSV file named 'df.csv' without writing row indices.
- df = pd.read_csv('df.csv'): This line reads the CSV file 'df.csv' into a new DataFrame df.
- df: Displays the DataFrame df.

Converting Salary Columns to Float:



```
[37]: df['Salary'] = df['Salary'].str.replace(',', '').str.replace('$', '').astype(float)
      df['Value'] = df['Value'].str.replace(',', '').str.replace('$', '').astype(float)
      df
```

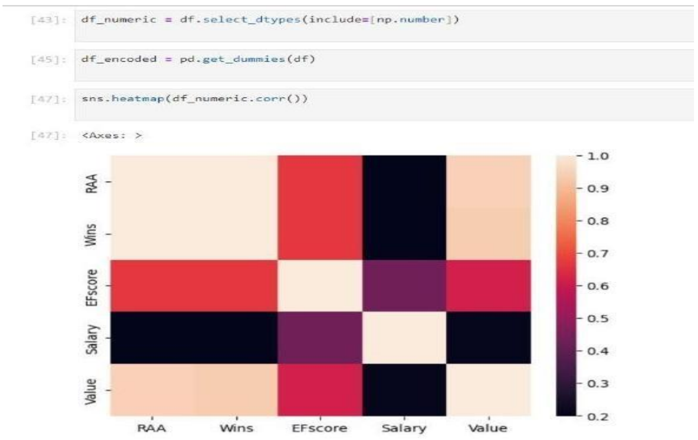
[37]:

	Player	Team	RAA	Wins	EFscore	Salary	Value
0	SE Marsh	Kings XI Punjab	377	1.355	0.231	0.0	1270355.0
1	SR Watson	Rajasthan Royals	347	1.250	0.297	125000.0	1209660.0
2	Sohail Tanvir	Rajasthan Royals	294	1.059	0.194	0.0	1099252.0
3	G Gambhir	Delhi Daredevils	216	0.776	0.160	725000.0	935664.0
4	GC Smith	Rajasthan Royals	202	0.725	0.139	250000.0	906183.0
...	...	...	...	...	...	...	...
2159	AR Patel	Delhi Capitals	-180	-0.607	0.081	714300.0	-297218.0
2160	NA Saini	Royal Challengers Bangalore	-191	-0.646	0.061	468750.0	-353939.0
2161	S Gopal	Rajasthan Royals	-215	-0.726	0.067	31250.0	-470290.0
2162	KH Pandya	Mumbai Indians	-251	-0.848	0.066	1375000.0	-647724.0
2163	SP Narine	Kolkata Knight Riders	-292	-0.988	0.051	1953130.0	-851338.0

2164 rows × 7 columns

Fig 4.4:Changing Datatype.

- df['Salary'] = df['Salary'].str.replace(',', '').str.replace('$', '').astype(float): This line in fig 4.4 removes commas anddollar signs from the 'Salary' column, then converts the column to float.
- df['Value'] = df['Value'].str.replace(',', '').str.replace('$', '').astype(float): This line removes commas and dollar signs from the 'Value' column, then converts the column to float.
- df: Displays the updated DataFrame df.

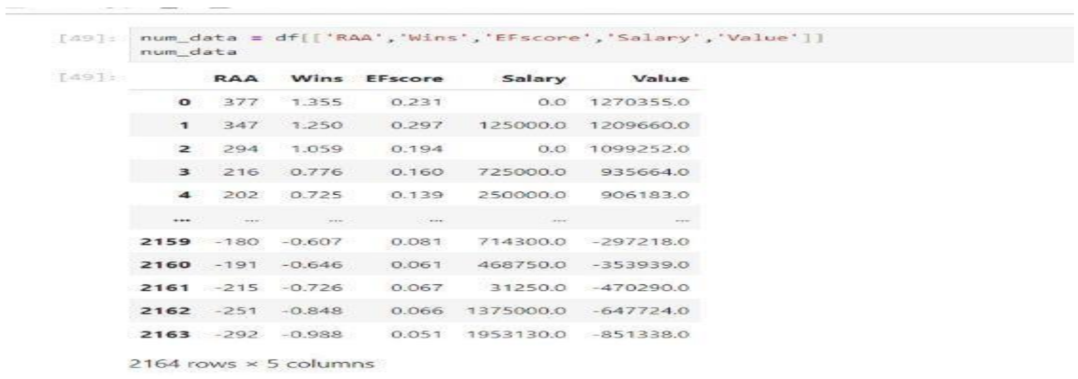


```
[43]: df_numeric = df.select_dtypes(include=[np.number])

[45]: df_encoded = pd.get_dummies(df)

[47]: sns.heatmap(df_numeric.corr())

[47]: <Axes: >
```

Fig 4.5:Correlation Of Heatmap.

Correlation Heatmap:

- df_numeric = df.select_dtypes(include=[np.number]): This line in fig 4.5 selects only the numericcolumns from the DataFrame df and assigns them to df_numeric.
- df_encoded = pd.get_dummies(df): This line converts categorical columns in the DataFrame df into dummy/indicator variables (one-hot encoding).
- sns.heatmap(df_numeric.corr()): This line creates a heatmap of the correlation matrix of the numeric columns in df_numeric using the Seaborn library.



```
[49]: num_data = df[['RAA','Wins','EFscore','Salary','Value']]
      num_data
```

[49]:

	RAA	Wins	EFscore	Salary	Value
0	377	1.355	0.231	0.0	1270355.0
1	347	1.250	0.297	125000.0	1209660.0
2	294	1.059	0.194	0.0	1099252.0
3	216	0.776	0.160	725000.0	935664.0
4	202	0.725	0.139	250000.0	906183.0
...	...	...	...	...	...
2159	-180	-0.607	0.081	714300.0	-297218.0
2160	-191	-0.646	0.061	468750.0	-353939.0
2161	-215	-0.726	0.067	31250.0	-470290.0
2162	-251	-0.848	0.066	1375000.0	-647724.0
2163	-292	-0.988	0.051	1953130.0	-851338.0

2164 rows × 5 columns

Fig 4.6 : Data Overview.

Data Overview and Data Types:

- num_data = df[['RAA', 'Wins', 'EFscore', 'Salary', 'Value']]: This line in fig 4.6 selects specific columns ('RAA', 'Wins', 'EFscore', 'Salary', 'Value') from the DataFrame df and assigns them to num_data.
- num_data: Displays the DataFrame num_data.
- num_data.info(): This line prints a concise summary of the DataFrame num_data, including the number of non-null entries and data types of each column.

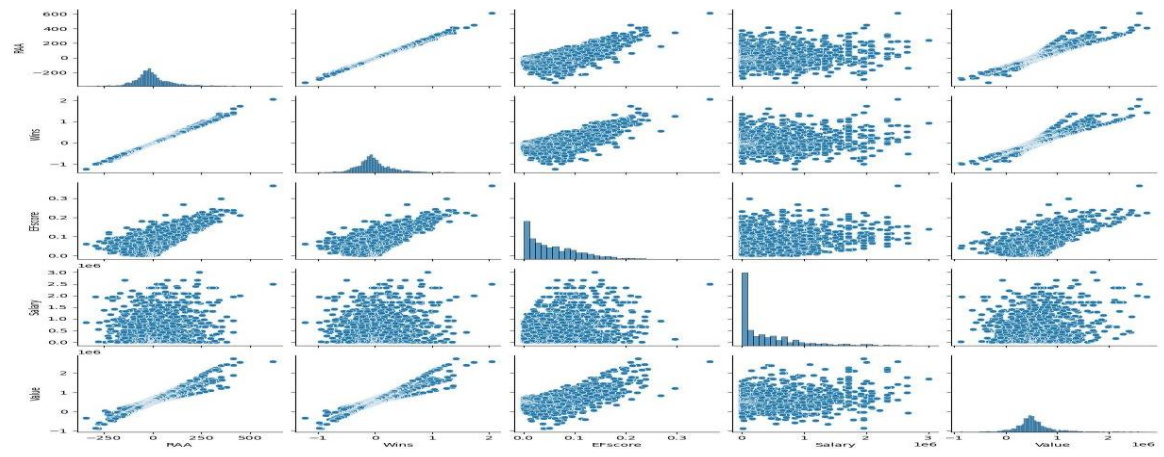


Fig 4.7:Scatter Plot.

This image in fig 4.7 shows a pairplot created using seaborn (sns.pairplot(num_data)). The pairplot visualizes relationships between multiple numerical variables in the dataset. Each variable is plotted against every other variable, creating a matrix of scatterplots. The diagonal shows the distribution of each variable.

The variables shown are RAA, Wins, EFscore, Salary, and Value. This visualization helps to identify correlations and patterns between these variables.

The pairplot allows for easy comparison of relationships between all pairs of variables. For example, you can see a strong positive correlation between RAA and Wins, and between Value and RAA/Wins.

In this Jupyter Notebook session, we worked on preparing a dataset for a machine learning project focused on predicting IPL player value. Here are the key steps we performed:

1. Data Cleaning:

- Removed the 'Rank' column from the dataset.

- Saved the cleaned DataFrame to a CSV file and then reloaded it for further processing.

2. Data Type Conversion:

- Converted the 'Salary' and 'Value' columns from string format (including commas and dollar signs) to float format for numerical analysis.

3. Exploratory Data Analysis (EDA):

- Selected only numeric columns for correlation analysis.

- Visualized the correlation matrix using a heatmap to understand the relationships between different numerical features.

4. Subset Selection: Extracted specific columns ('RAA', 'Wins', 'EFscore', 'Salary', 'Value') for focused analysis.

- Displayed a summary of the selected subset, including data types and non-null entries.

How Jupyter Notebook Helped Us:

1. Interactive Environment: - Allowed us to write and execute code in a step-by-step manner, making it easy to test and debug.

- Enabled immediate feedback by displaying outputs directly below the code cells.

2. Data Visualization:

- Provided support for integrating visualization libraries like Seaborn, enabling us to create visual representations of data (e.g., correlation heatmap) to gain insights quickly.

3. Documentation and Narration:

- Allowed us to add markdown cells for documentation, explaining each step and its purpose, making the notebook a self-contained and comprehensible document.

4. Reproducibility:

- Made it easy to save and reload datasets, ensuring that the workflow can be reproduced shared with others.

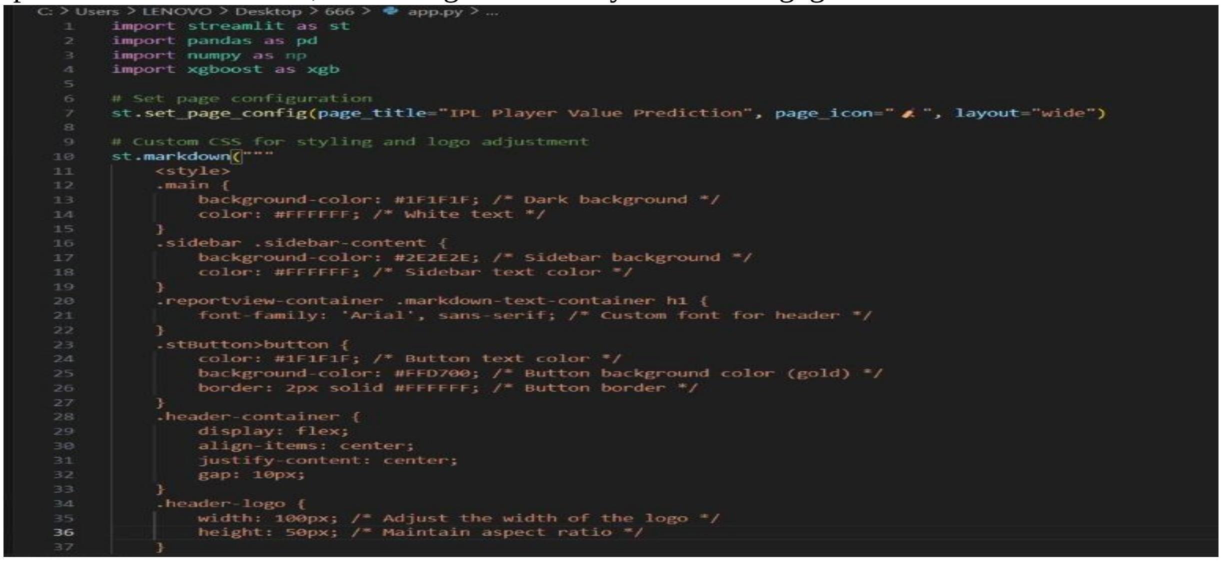
5. Code and Output Together:

- Displayed both the code and its corresponding output in a single interface, making it easier to understand the impact of each operation on the dataset.

Overall, Jupyter Notebook's interactive and user-friendly interface significantly enhanced our ability to manipulate, analyze, and visualize data efficiently throughout this project.

Transforming Machine Learning Projects into Interactive Web Apps with Streamlit:

Streamlit is an open-source framework that enables machine learning developers to create interactive web applications quickly. It allows for easy integration of Python scripts and data visualizations, making it ideal for showcasing machine learning models and their results. By using Streamlit, the code from the Jupyter Notebook can be transformed into a user-friendly web app, enabling users to interact with data visualizations, input parameters, and view model predictions in real-time, enhancing accessibility and user engagement.



```
C: > Users > LENOVO > Desktop > 666 > app.py > ...
import streamlit as st
import pandas as pd
import numpy as np
import xgboost as xgb

# Set page configuration
st.set_page_config(page_title="IPL Player Value Prediction", page_icon=" ", layout="wide")

# Custom CSS for styling and logo adjustment
st.markdown("""
    <style>
    .main {
        background-color: #1F1F1F; /* Dark background */
        color: #FFFFFF; /* White text */
    }
    .sidebar .sidebar-content {
        background-color: #2E2E2E; /* Sidebar background */
        color: #FFFFFF; /* Sidebar text color */
    }
    .reportview-container .markdown-text-container h1 {
        font-family: 'Arial', sans-serif; /* Custom font for header */
    }
    .stButton>button {
        color: #1F1F1F; /* Button text color */
        background-color: #FFD700; /* Button background color (gold) */
        border: 2px solid #FFFFFF; /* Button border */
    }
    .header-container {
        display: flex;
        align-items: center;
        justify-content: center;
        gap: 10px;
    }
    .header-logo {
        width: 100px; /* Adjust the width of the logo */
        height: 50px; /* Maintain aspect ratio */
    }
```

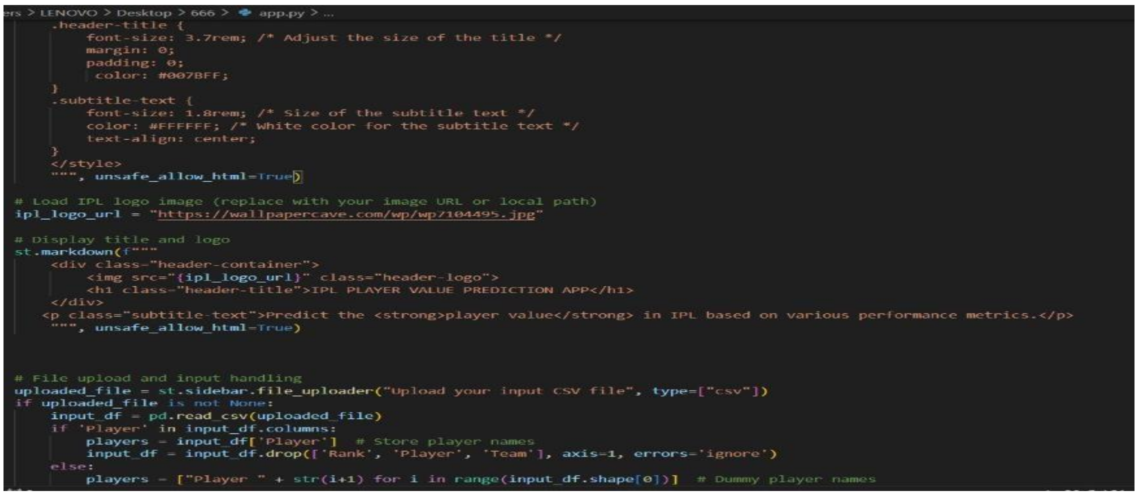
Fig 4.8:Importing Libraries.

This fig 4.8 shows the beginning of a Streamlit application for IPL Player Value Prediction.

1. Import statements for required libraries: streamlit, pandas, numpy, and xgboost.
2. Streamlit page configuration is set with a title, icon, and layout.

3. Custom CSS is defined for styling the application, including:
   - Dark background and white text for the main content
   - Sidebar styling
   - Custom font for headers
   - Button styling with gold background
   - Flexbox layout for the header container
   - Logo size adjustment

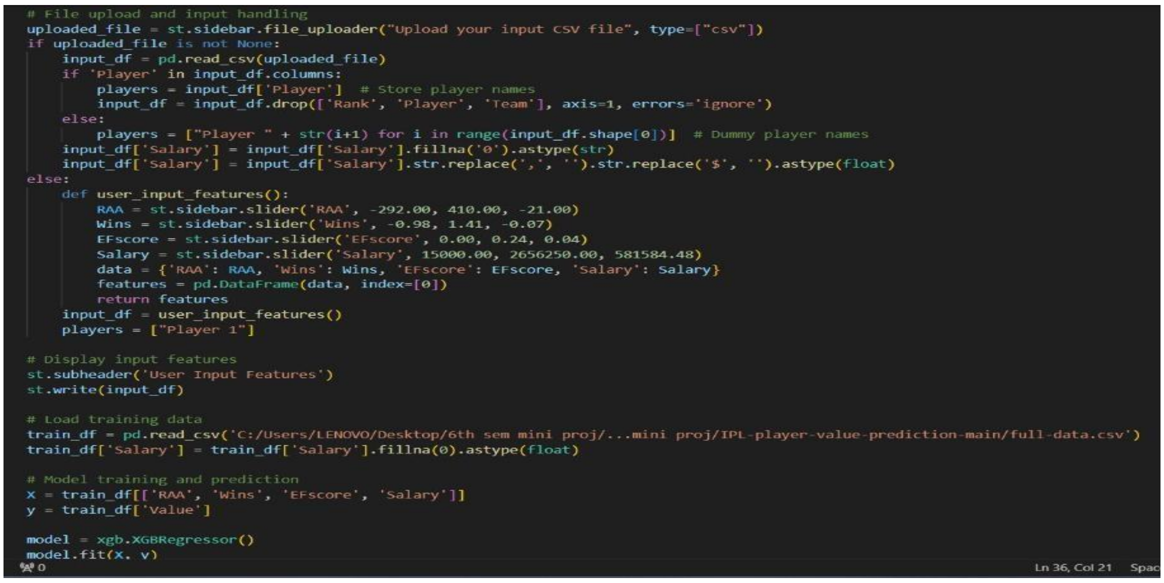
This image continues the CSS styling and starts the main application content.



```
ers > LENOVO > Desktop > 666 > app.py > ...
    .header-title {
        font-size: 3.7rem; /* Adjust the size of the title */
        margin: 0;
        padding: 0;
         color: #007BFF;
    }
    .subtitle-text {
        font-size: 1.8rem; /* Size of the subtitle text */
        color: #FFFFFF; /* White color for the subtitle text */
        text-align: center;
    }
    </style>
    """, unsafe_allow_html=True)

# Load IPL logo image (replace with your image URL or local path)
ipl_logo_url = "https://wallpapercave.com/wp/wp7104495.jpg"

# Display title and logo
st.markdown(f"""
    <div class="header-container">
        <img src="{ipl_logo_url}" class="header-logo">
        <h1 class="header-title">IPL PLAYER VALUE PREDICTION APP</h1>
    </div>
   <p class="subtitle-text">Predict the <strong>player value</strong> in IPL based on various performance metrics.</p>
    """, unsafe_allow_html=True)


# File upload and input handling
uploaded_file = st.sidebar.file_uploader("Upload your input CSV file", type=["csv"])
if uploaded_file is not None:
    input_df = pd.read_csv(uploaded_file)
    if 'Player' in input_df.columns:
        players = input_df['Player']  # Store player names
        input_df = input_df.drop(['Rank', 'Player', 'Team'], axis=1, errors='ignore')
    else:
        players = ["Player " + str(i+1) for i in range(input_df.shape[0])]  # Dummy player names
```

Fig 4.9:CSS Section.

1. Additional CSS for header title and subtitle text [in fig 4.9]
2. Loading of an IPL logo image from a URL
3. Displaying the title and logo using Streamlit's markdown function
4. File upload functionality for CSV input
5. If a file is uploaded:
   - Read the CSV file into a pandas DataFrame
   - Extract player names if available
   - Drop unnecessary columns (Rank, Player, Team)
6. If no file is uploaded, create dummy player names.

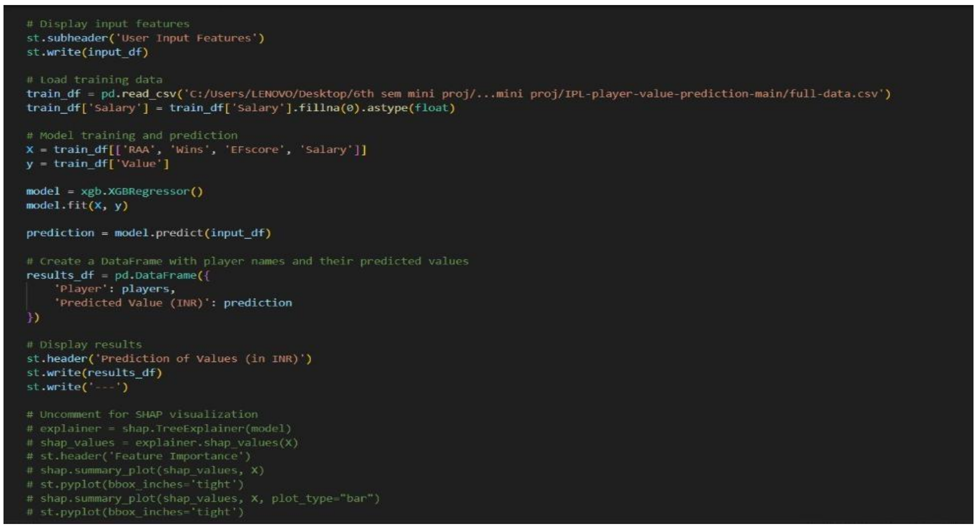


```
# File upload and input handling
uploaded_file = st.sidebar.file_uploader("Upload your input CSV file", type=["csv"])
if uploaded_file is not None:
    input_df = pd.read_csv(uploaded_file)
    if 'Player' in input_df.columns:
        players = input_df['Player']  # Store player names
        input_df = input_df.drop(['Rank', 'Player', 'Team'], axis=1, errors='ignore')
    else:
        players = ["Player " + str(i+1) for i in range(input_df.shape[0])]  # Dummy player names
    input_df['Salary'] = input_df['Salary'].fillna('0').astype(str)
    input_df['Salary'] = input_df['Salary'].str.replace(',', '').str.replace('$', '').astype(float)
else:
    def user_input_features():
        RAA = st.sidebar.slider('RAA', -292.00, 410.00, -21.00)
        Wins = st.sidebar.slider('Wins', -0.98, 1.41, -0.07)
        EFscore = st.sidebar.slider('EFscore', 0.00, 0.24, 0.04)
        Salary = st.sidebar.slider('Salary', 15000.00, 2656250.00, 581584.48)
        data = {'RAA': RAA, 'Wins': Wins, 'EFscore': EFscore, 'Salary': Salary}
        features = pd.DataFrame(data, index=[0])
        return features
    input_df = user_input_features()
    players = ["Player 1"]

# Display input features
st.subheader('User Input Features')
st.write(input_df)

# Load training data
train_df = pd.read_csv('C:/Users/LENOVO/Desktop/6th sem mini proj/...mini proj/IPL-player-value-prediction-main/full-data.csv')
train_df['Salary'] = train_df['Salary'].fillna(0).astype(float)

# Model training and prediction
X = train_df[['RAA', 'Wins', 'EFscore', 'Salary']]
y = train_df['Value']

model = xgb.XGBRegressor()
model.fit(X, y)
```

Fig 4.10:Data Preprocessing And Feature Handling.

This fig 4.10 shows the data preprocessing and feature input handling.

1. Cleaning the salary data by replacing commas and dollar signs, converting to float
2. Definition of a function user_input_features() that creates sliders for user input:
    - RAA (Runs Above Average)
    - Wins
    - EFscore (Efficiency Score)
    - Salary
3. Creation of a DataFrame from user inputs
4. Loading of training data from a CSV file
5. Cleaning the salary data in the training set
6. Preparing features (X) and target variable (y) for model training.



```
# Display input features
st.subheader('User Input Features')
st.write(input_df)

# Load training data
train_df = pd.read_csv('C:/Users/LENOVO/Desktop/6th sem mini proj/...mini proj/IPL-player-value-prediction-main/full-data.csv')
train_df['Salary'] = train_df['Salary'].fillna(0).astype(float)

# Model training and prediction
X = train_df[['RAA', 'Wins', 'EFscore', 'Salary']]
y = train_df['Value']

model = xgb.XGBRegressor()
model.fit(X, y)

prediction = model.predict(input_df)

# Create a DataFrame with player names and their predicted values
results_df = pd.DataFrame({
    'Player': players,
    'Predicted Value (INR)': prediction
})

# Display results
st.header('Prediction of Values (in INR)')
st.write(results_df)
st.write('---')

# Uncomment for SHAP visualization
# explainer = shap.TreeExplainer(model)
# shap_values = explainer.shap_values(X)
# st.header('Feature Importance')
# shap.summary_plot(shap_values, X)
# st.pyplot(bbox_inches='tight')
# shap.summary_plot(shap_values, X, plot_type="bar")
# st.pyplot(bbox_inches='tight')
```

Fig 4.11:Model Training.

This fig 4.11 shows the model training, prediction, and results display.

1. Creation and training of an XGBoost regression model
2. Making predictions using the trained model on user input
3. Creating a DataFrame with player names and their predicted values
4. Displaying the prediction results using Streamlit
5. Commented-out code for SHAP (SHapley Additive exPlanations) visualization, which could be used to explain feature importance

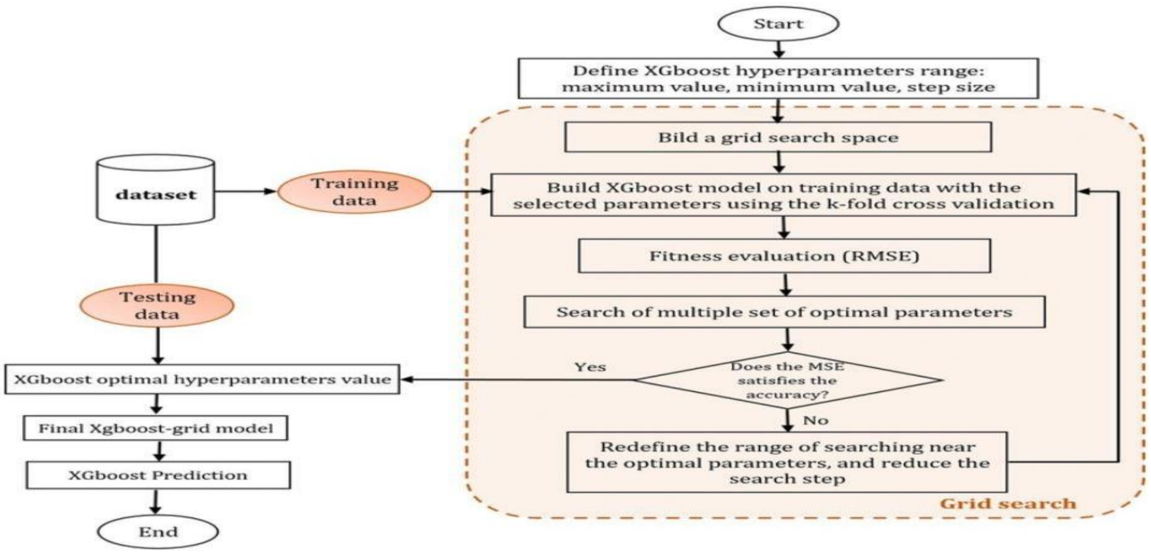
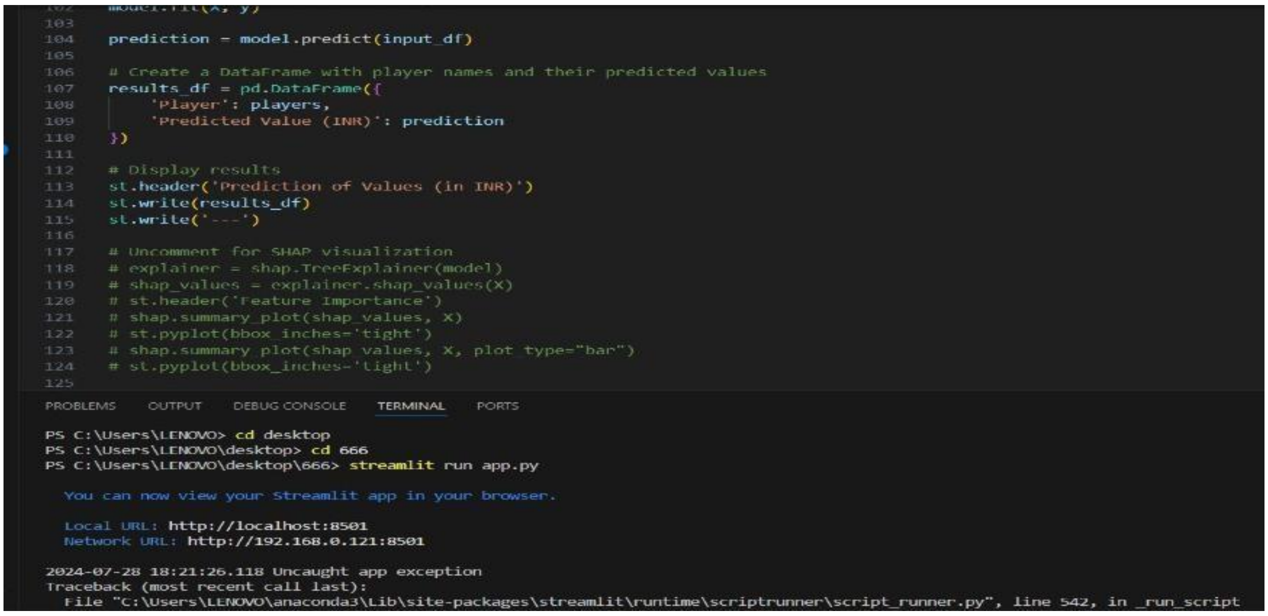

Start
Define XGboost hyperparameters range: maximum value, minimum value, step size
Bild a grid search space
dataset
Training data
Build XGboost model on training data with the selected parameters using the k-fold cross validation
Fitness evaluation (RMSE)
Testing data
Search of multiple set of optimal parameters
Yes
Does the MSE satisfies the accuracy?
XGboost optimal hyperparameters value
No
Final Xgboost-grid model
Redefine the range of searching near the optimal parameters, and reduce the search step
XGboost Prediction
Grid search
End

Fig 4.12:XGB Model Methodology.

Overall, this code in fig 4.12 creates a Streamlit web application that allows users to input player statistics and predicts the player's value using an XGBoost regression model[10] trained on historical IPL data. The application includes custom styling, file upload capabilities, and interactive input features.

CHAPTER 5

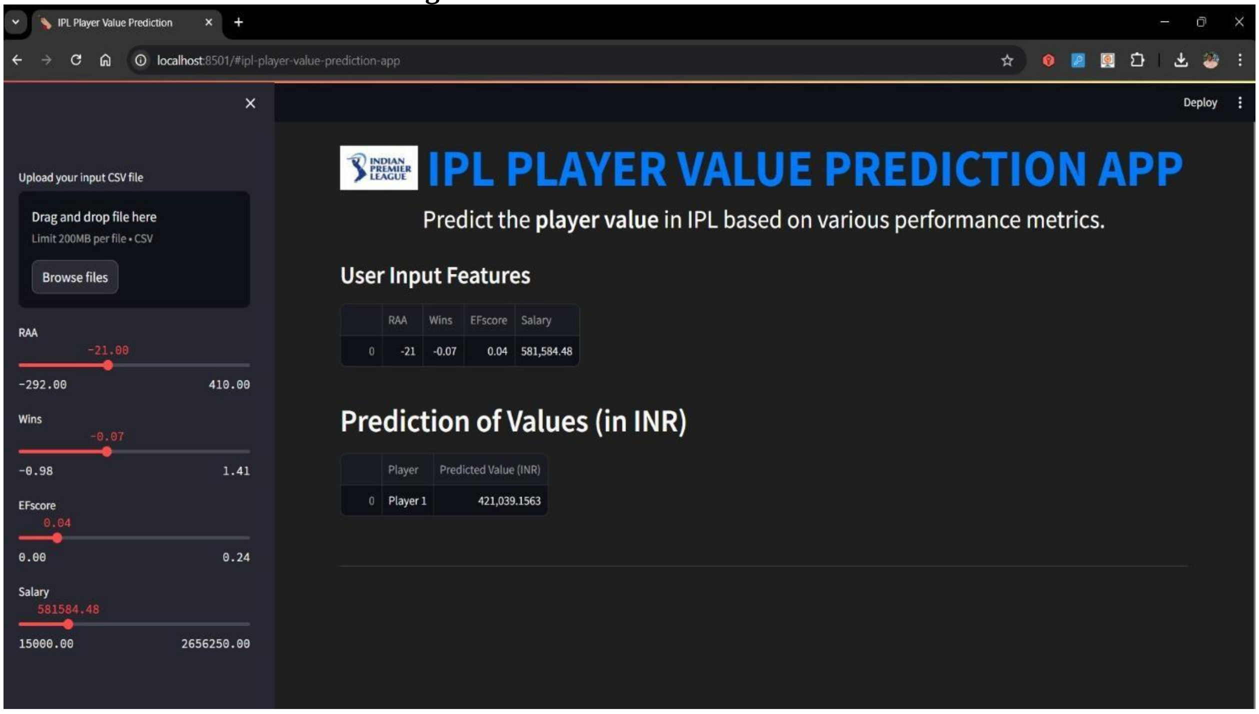
OUTPUT:



```
model.fit(X, y)

prediction = model.predict(input_df)

# Create a DataFrame with player names and their predicted values
results_df = pd.DataFrame({
    'Player': players,
    'Predicted Value (INR)': prediction
})

# Display results
st.header('Prediction of Values (in INR)')
st.write(results_df)
st.write('---')

# Uncomment for SHAP visualization
# explainer = shap.TreeExplainer(model)
# shap_values = explainer.shap_values(X)
# st.header('Feature Importance')
# shap.summary_plot(shap_values, X)
# st.pyplot(bbox_inches='tight')
# shap.summary_plot(shap_values, X, plot_type="bar")
# st.pyplot(bbox_inches='tight')
```

```
PS C:\Users\LENOVO> cd desktop
PS C:\Users\LENOVO\desktop> cd 666
PS C:\Users\LENOVO\desktop\666> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.121:8501

2024-07-28 18:21:26.118 Uncaught app exception
Traceback (most recent call last):
  File "C:\Users\LENOVO\anaconda3\Lib\site-packages\streamlit\runtime\scriptrunner\script_runner.py", line 542, in _run_script
```

Fig 5.1:Streamlit Run Command.



IPL Player Value Prediction
localhost:8501/#ipl-player-value-prediction-app
Deploy
INDIAN PREMIER LEAGUE

Upload your input CSV file

Drag and drop file here
Limit 200MB per file • CSV
Browse files

RAA -21.00 (-292.00 – 410.00)

Wins -0.07 (-0.98 – 1.41)

EFscore 0.04 (0.00 – 0.24)

Salary 581584.48 (15000.00 – 2656250.00)

IPL PLAYER VALUE PREDICTION APP

Predict the **player value** in IPL based on various performance metrics.

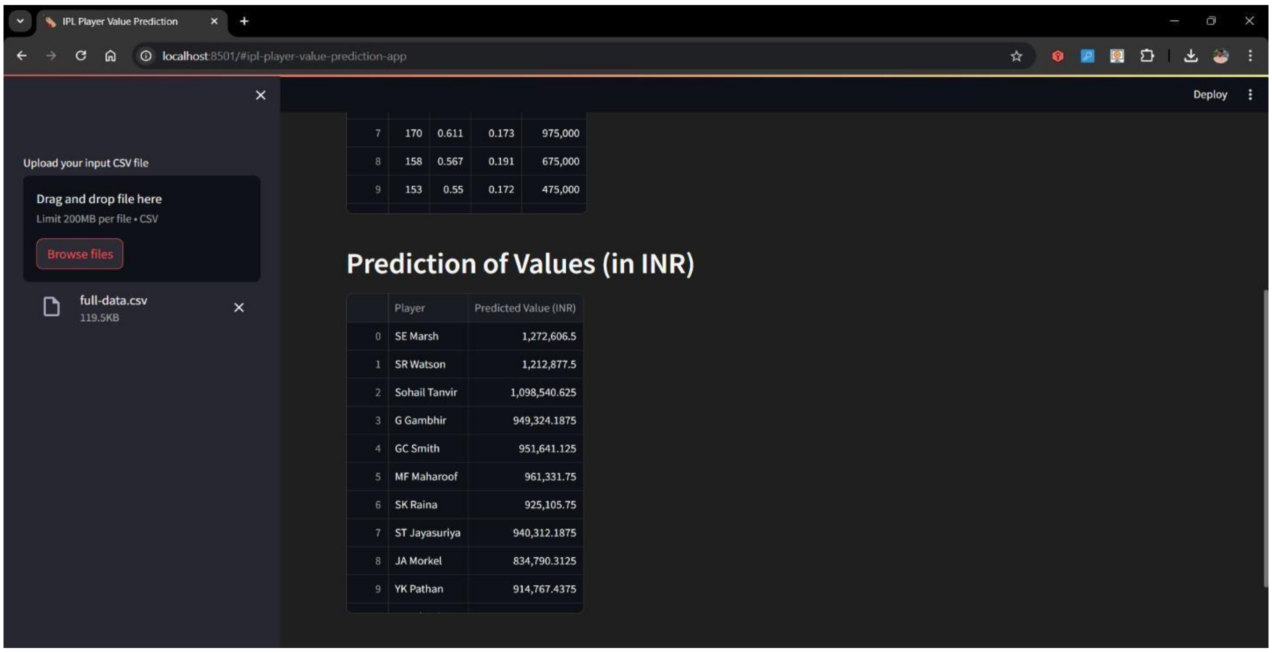
User Input Features

	RAA	Wins	EFscore	Salary
0	-21	-0.07	0.04	581,584.48

Prediction of Values (in INR)

	Player	Predicted Value (INR)
0	Player 1	421,039.1563

Fig 5.2:Output Webpage.

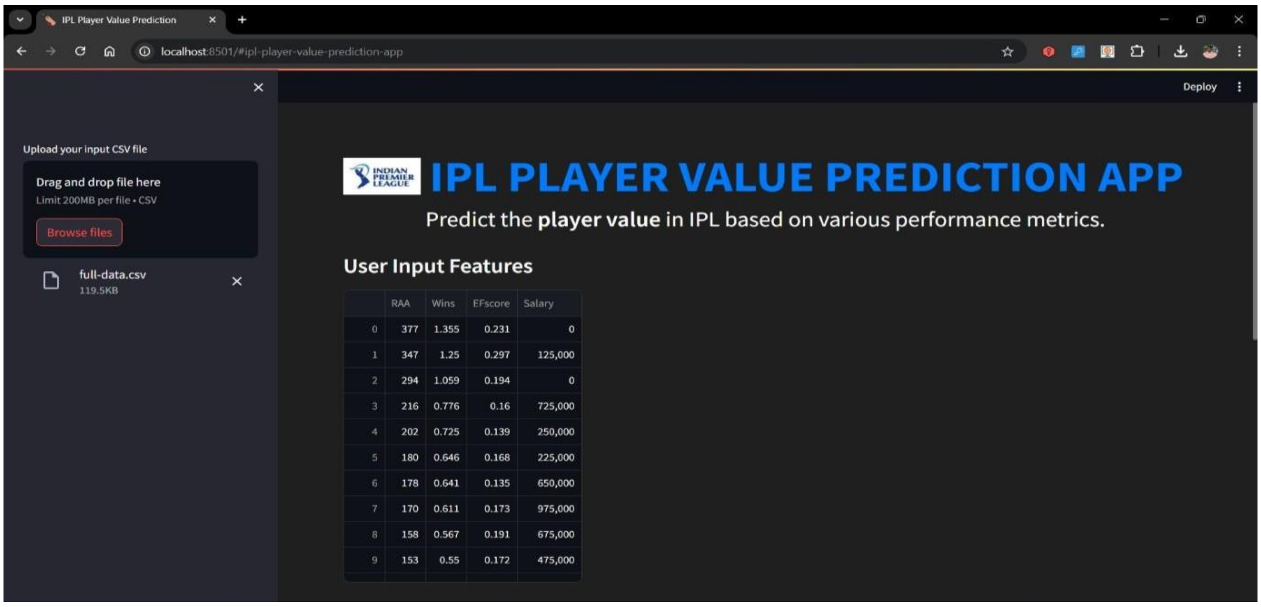

IPL Player Value Prediction
localhost:8501/#ipl-player-value-prediction-app
Deploy
Upload your input CSV file
Drag and drop file here
Limit 200MB per file • CSV
Browse files
full-data.csv
119.5KB

7	170	0.611	0.173	975,000
8	158	0.567	0.191	675,000
9	153	0.55	0.172	475,000

Prediction of Values (in INR)

	Player	Predicted Value (INR)
0	SE Marsh	1,272,606.5
1	SR Watson	1,212,877.5
2	Sohail Tanvir	1,098,540.625
3	G Gambhir	949,324.1875
4	GC Smith	951,641.125
5	MF Maharoof	961,331.75
6	SK Raina	925,105.75
7	ST Jayasuriya	940,312.1875
8	JA Morkel	834,790.3125
9	YK Pathan	914,767.4375

Fig 5.3:Importing Datasets for Prediction.


IPL Player Value Prediction
localhost:8501/#ipl-player-value-prediction-app
Deploy
Upload your input CSV file
Drag and drop file here
Limit 200MB per file • CSV
Browse files
full-data.csv
119.5KB
INDIAN PREMIER LEAGUE

IPL PLAYER VALUE PREDICTION APP

Predict the **player value** in IPL based on various performance metrics.

User Input Features

	RAA	Wins	EFscore	Salary
0	377	1.355	0.231	0
1	347	1.25	0.297	125,000
2	294	1.059	0.194	0
3	216	0.776	0.16	725,000
4	202	0.725	0.139	250,000
5	180	0.646	0.168	225,000
6	178	0.641	0.135	650,000
7	170	0.611	0.173	975,000
8	158	0.567	0.191	675,000
9	153	0.55	0.172	475,000

Fig 5.4:Value Predicted.

This Streamlit application is designed for IPL (Indian Premier League) Player Value Prediction. Here's a detailed explanation of the code and functionality shown across the four images:

1. Setup and Styling:

- The app imports necessary libraries: streamlit, pandas, numpy, and xgboost.
- It sets the page configuration in fig 5.1 with the title "IPL Player Value Prediction", a cricket ballicon, and a wide layout.
- Custom CSS is defined to style the application:
  - Dark background (#1F1F1F) with white text for the main content
  - Sidebar styling with a different background color (#2E2E2E)
  - Custom font (Arial) for headers
  - Button styling with gold background (#FED700) and white border
  - Flexbox layout for the header container
  - Logo size adjustment (100px width, 50px height)

2. Header and File Upload:

- The app loads an IPL logo from a URL.
- It displays the title "IPL PLAYER VALUE PREDICTION APP" and the logo using Streamlit's markdown function.
- A subtitle in fig 5.2 explains the app's purpose: "Predict the player value in IPL based on variousperformance metrics."
- File upload functionality is implemented for CSV input.
- If a file is uploaded:
  - The CSV is read into a pandas DataFrame.
  - Player names are extracted if available.
  - Unnecessary columns (Rank, Player, Team) are dropped.
- If no file is uploaded, dummy player names are created.

3. Data Preprocessing and User Input :

- A function `user_input_features()` in fig 5.3 is defined to create sliders for user input:
  - RAA (Runs Above Average): range -292.00 to 410.00
  - Wins: range -0.98 to 1.41
  - EFscore (Efficiency Score): range 0.00 to 0.24

- Salary: range 15000.00 to 2656250.00

- These inputs are combined into a DataFrame.

- The app loads training data from a CSV file and cleans the salary data.

- Features (X) and target variable (y) are prepared for model training.

4. Model Training and Prediction:

- An XGBoost regression model in fig 5.4 is created and trained on the data.

- The model makes predictions using the user input.

- Results are formatted into a DataFrame with player names and predicted values.

- The app displays the prediction results using Streamlit's `write` function.

- There's commented-out code for SHAP (SHapley Additive exPlanations) visualization, which could be used to explain feature importance.

Overall, this Streamlit application provides a user-friendly interface for predicting IPL player values based on performance metrics. Users can either upload their own data or use the sliders to input player statistics, and the app will use an XGBoost model to predict the player's value. The application is styled to look professional and theme-appropriate for cricket, with a dark background and gold accents.

CHAPTER 6

CONCLUSION

In this machine learning project, we aimed to predict the value of IPL players based on their performance metrics. The player value was determined by their Runs Above Average (RAA) scores, which combined both batting and bowling contributions. An RAA score of 0 indicated an average performance, while positive and negative scores represented above-average and below-average performances, respectively. The player valuation methodology redistributed the total team spending based on individual player performances.

We employed the 'Extreme Gradient Boosting' regression model to tackle this prediction problem, using parameters such as RAA, EFscore, Wins, and Salary. Using Jupyter Notebook, we effectively visualized data, performed data cleaning, and conducted exploratory data analysis. The insights gained from these analyses were crucial in understanding the data and refining our model.

Furthermore, we leveraged Visual Studio Code and Streamlit to develop an interactive web application, allowing users to engage with the model's predictions and data visualizations in real time. This project not only demonstrated the practical application of machine learning techniques but also highlighted the importance of integrating various tools for data analysis and web development to create a comprehensive and user-friendly solution.

REFERENCES

[1]www.iplt20.com

[2]stats.espncricinfo.com

[3]"The Use of Statistics in Cricket" by S. Raghunathan, Journal of Sports Analytics,2018

[4]"Machine Learning Approaches to Cricket Score Prediction" by M. Patel et al., IEEEConference on Sports Analytics, 2020

[5]Scikit-learn Documentation: scikit-learn.org

[6]Streamlit Documentation: docs.streamlit.io.

[7]Srikantaiah, K. C., Khetan, A., Kumar, B., Tolani, D., & Patel, H. (2021). Prediction of IPL Match Outcome Using Machine Learning Techniques. Proceedings of the 3rd International Conference on Integrated Intelligent Computing Communication & Security (ICIIC 2021).

[8]Mohapatra, S., Goswami, A., Singh, A., Singh, V. K., Sen, B., & Sharma, K. (2022). Exploratory Data Analysis on IPL Data. Lecture Notes in Electrical Engineering, 750, 417-424.

[9]http://www.cricmetric.com/index.py#google_vignette

[10]https://www.geeksforgeeks.org/xgboost/

[11]https://www.imsl.com/blog/what-is-regression-model#:~:text=A%20regression%20model%20provides%20a,by%20a%20linear%20regression%20model.

[12]https://www.analyticsvidhya.com/blog/2023/06/the-science-of-t20-cricket-decoding- player-performance-with-predictive-modeling/

[13]S. Khare and P. Saxena, "Performance Evaluation and Prediction in T20 Cricket Using Machine Learning," International Journal of Sports Science and Performance Analysis, vol. 20, no. 1, pp. 112-124, 2020.

[14]S. Sinha and S. Chakraborty, "Application of Machine Learning in Predicting Cricket Player Performance: A Review," International Journal of Sports Science & Coaching, vol. 16, no. 6, pp. 1423-1431, 2021.