International Centre for Education and Research (ICER)
VIT-Bangalore

IPL AUCTION PREDICTION MODEL USING

MACHINE LEARNING TECHNIQUES

CS6510 – PROJECT 1REPORT

Submitted by

G.NITHESH CHOWDARY – 23MSP3017

In partial fulfillment for the award of the degree of

POST GRADUATE PROGRAMME

INTERNATIONAL CENTRE FOR HIGHER EDUCATION AND RESEARCHVIT

BANGALORE

DECEMBER, 2023

International Centre for Education and Research (ICER)
VIT-Bangalore

BONAFIDE CERTIFICATE

Certified that this project report **"IPL Auction Prediction Model using Machine Learning Techniques"** is the bonafide record of work done by **"G.Nithesh Chowdary – 23MSP3017"** who carried out the project work under my supervision.

Signature of the Supervisor

Prof.Ramya Mohanakrishnan

Assistant Professor,

ICER

VIT Bangalore

Signature of Director

Prof. Prema M

Director,

ICER

VIT Bangalore.

Evaluation Date:

International Centre for Education and Research (ICER)
VIT-Bangalore

ACKNOWLEDGEMENT

I express my sincere gratitude to our director of ICER **Prof. Prema M.** for their support and for providing the required facilities for carrying out this study.

I wish to thank my faculty supervisor(s), **Prof.Ramya Mohanakrishnan**, **Assistant Professor** ICER for extending help and encouragement throughout the project. Without his/her continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to all the members of ICER, my beloved parents, and friends for extending the support, who helped us to overcome obstacles in the study.

TABLE OF CONTENTS

ABSTRACT

The Indian Premier League (IPL) auction serves as a pivotal event where cricket franchisesbid on players to assemble competitive teams. The goal of this work is to construct an IPL auction prediction model that can reliably predict player prices. By utilizing past player performance information, team specifications, contextual data, and cutting-edge machine learning methods, the model aims to provide useful information to teams taking part in the auctions. The dataset is obtained from Kaggle which consists of the player's previous statistics, Auction sold price, base price, and player details. A machine learning model withLinear Regression, Random Forest, and gradient boosting algorithms will be implemented and the accuracy is measured by Mean Squared Error (MSE) and Root Mean SquaredError (RMSE). Finally, to find the best model we will compare the accuracy of the models. The expected results include accurate player price forecasts, supporting teams in allocatingfunds as efficiently as possible, enabling knowledgeable decision-making during auctions, and maybe enhancing team lineups for increased IPL competition. The model'sdeployment and validation against actual auction prices aim to validate its efficacy and contribute to the landscape of sports analytics.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. INDIAN PREMIER LEAGUE (IPL)

One of the world's most thrilling and popular cricket competitions, the Indian Premier League (IPL) embodies the spirit of world-class cricket, entertainment, andintense rivalry. The Indian Premier League (IPL), which was introduced in 2008 bythe Board of Control for Cricket in India (BCCI), transformed the game by combining elite cricket players with an enthralling spectacle of entertainment and fan interaction.

The Indian Premier League (IPL) is an exciting blend of glamourous entertainment and skillful cricket. It features a novel concept where franchise-based teams, each representing a city, compete in a high-stakes Twenty20 (T20) cricket tournament over a number of weeks. Famous for its intense matches, star-studded lineups, and thrilling competitions, the IPL has carved out a position for itself by drawing awide international audience that cuts over national and cricketing allegiances.

1.2. STATISTICAL INFORMATION

The IPL auction data spanning from 2013 to 2023 reveals a significant evolution in player acquisitions and spending over the years. In 2013, 37 players were sold, amounting to a total expenditure of 11.89 million. The following years witnessed a remarkable surge in both the number of players sold and the total spending. In2014, a staggering 154 players were acquired, marking an expenditure of 2.6billion. The trend continued in 2015 with 67 players sold for 876 million and in 2016 with 94 players at 136 million. Notably, 2017 saw 66 players being sold for a whopping 911 million. The subsequent years displayed a substantial increase in both player count and spending, with 169 players in 2018 for 4.3 billion, 60 playersin 2019 for 1.06 billion, 62 players in 2020 for 1.4 billion, 57 players in 2021 for 1.45 billion, and a remarkable escalation in 2022 with 204 players sold for 5.5 billion. The latest data from 2023 indicated 80 players were sold for 1.67 billion, reflecting a consistently high investment in player acquisitions, underlining the escalating commercial value and stakes associated with the IPL auctions.

CHAPTER 2
LITERATURE REVIEW

This section talks about the other previous works on prediction of Player price in ipl auction.

The journal [1] uses an approach focused on explainable Machine Learning (ML) techniques for the prediction of player valuation in IPL auctions. Our methodology emphasizes transparency and interpretability in the model's decision-making process. The final outcomes are depicted through explanatory visualizations, offering insights into the crucial metrics influencing player selection. Figure 2 delineates individual player cases, highlighting key metrics that significantly influence their chances of being picked. For instance, in Case 1, the positive relationship between a player's teamand reserve price emerges as influential, while age exhibits a negative impact on the player's likelihood of being chosen. Figure 3 consolidates these individual metric explanations, providing a comprehensive overview of the combined importance attributed to various metrics concerning player selection, thereby offering a holistic understanding of the factors shaping player valuation in IPL auctions.

In Journal [2], has produced a new model designed for precise IPL player valuation by utilizing a larger dataset that includes a variety of cricket formats and seasons. This research thoroughly examines the operational effectiveness ofthe IPL Auctions by means of a comprehensive examination of surplus and deficit players. The results illustrate the effectiveness of data-driven approaches, and the created Linear Regression machine learning algorithm's predictive power is especially noteworthy. Interestingly, the algorithm highlights significant discrepancies between player performance in real life and their auction values, illuminating the shortcomings in the IPL auction system's valuation. These disclosures highlight the key importance of this study and its overall goals.

Paper [3] tackles the critical challenge of determining base prices for players in an IPL mega auction, aiming to streamline the auction process by optimizing base price suggestions. The approach pivots towards enhancing algorithmic accuracy to minimize discrepancies between suggested and actual auction prices. Demonstrating the effectiveness of the proposed methodology, the results indicate an impressive accuracy of approximately 74% for Indian players and around 58% for Foreign players in predicting their respective base prices. Furthermore, the algorithm significantly streamlines the auction process, reducing the time taken by approximately 17.6% for Indian players and about 31.1% for Foreign players. Notably, the analysis of the2022 mega auction data corroborates the outcomes observed during the 2018 mega auction, reinforcing the robustness and consistency of the developed approach across different IPL auction seasons.

paper [4] presents a novel model that reveals the cricketers' price valuation inthe most recent IPL auctions by combining the strong XGBoost Learning technique with the modified hedonic pricing model. Through the integration ofmany approaches, the model skillfully navigates the complexities involved in choosing traits that are essential for predicting player pricing, providing bidders with an advanced tool for effective player valuation and bidding tactics. With the goal of empowering bidders and enabling the best possible investment choices based on a player's performance potential, the suggestedmodel aims to improve efficiency and eliminate mistakes in the player acquisition process during IPL auctions.

Research paper [5] suggested SVR and linear regression models have skillfully integrated necessary characteristics to accurately forecast the costs of bowlers and batters. It is noteworthy that the pricing system takes into consideration the performance trajectory, meaning that players with strong performances are valued more highly, while those with poorer performances are valued less highly. With bowlers' average error of ±₹4,50,000 and batsmen's average error of ±₹5,20,000, the models' accuracy is impressive. 8% of the ₹80,00,00,000 budget is acceptable, even in the worst-case scenario if the auctioneer bids for 22 bowlers, potentially resulting in a ₹6,00,00,000 miscalculation in the budget.

Research paper [6] used the Multiple Linear Regression approach to model the intricate link between several dependent variables (Base Price Score) and an independent variable. The fact that there were several dependent factors linked to the independent variable made this decision especially appropriate. Additionally, the Root Mean Square Error approach was included to test the prediction accuracy of the model, offering a reliable way to evaluate how well the model performed in projecting player base values for the IPL Auction.

Research paper [7] findings highlight the critical elements affecting a cricket player's bidding value, identifying star value, nationality, and the three unique cricketing variables as the primary drivers. Compared to the three models that came before it, this one is distinguished by its ease of use and increased accuracy in estimating a player's worth throughout the bidding process. The focus on these core factors represents a break from more complex approaches and confirms the usefulness of a simple but effective strategy for determining a cricketer's value in bidding situations.

Research paper [8] This article discusses a crucial technique for ranking players, recommending the best cricket team, and projecting match results. Recent playerperformance, career statistics overall, performance against various opponents on various grounds, Day/Night, Home/Away, and Batting First/Second conditions are theprimary factors taken into account for both cricket squad recommendation and result prediction. Based on the computed batting, bowling, and fielding scores, players are chosen for the squad. Scores are determined by projecting each player's performance in the upcoming match. In addition to future improvements in hard-hitting, finishing, and running between wickets, overall career stats are taken into account for both batting and bowling scores. Additionally, total dot balls bowled, total extra runsconceded, and total boundaries conceded are taken into account. The authors discovered that wicket keepers have higher weights and should be separated from fielders as a result of a calculated fielder weight comparison. Because ground and opponent data are used to predict outcomes, players who haven't played in a particular ground against a specific opponent may be overlooked, which must be addressed inthe future.

Research paper [9] paper's primary goal is to develop a model for forecasting the first innings' final score and the second innings' match result for one-day internationals (ODIs). Using the Linear Regression and Naïve Bayes classifiers on previous ODI matches, two distinct models have been proposed, one for the first innings and anotherfor the second. By analyzing the actual ODI cricket data, the current method for projecting the score and the error of the linear regression classifier predictions were compared. It was found that the Linear Regression Classifier predicts the final score ofthe match with less error than the Current Run Rate method in all scenarios. Additionally, as the match goes on, the Naïve Bayes prediction accuracy increases from 70% (at first) to 91%. The goal going forward will be to raise both models' accuracy levels. Predictions will also take into account additional elements such as home team advantage, the toss, and the teams' ODI rankings.

Research paper [10] reveals a strong relationship between the prices at which players are auctioned off and their performance in IPL games. Interestingly, there is a strong positive association found between auction prices and players' batting and bowling totals. Furthermore, our analysis reveals the critical role that player performances havein the overall success of IPL teams. By using the developed prediction model, we wereable to predict the IPL season winners with high accuracy using team batting and bowling averages.

CHAPTER 3
OBJECTIVE

To develop and validate predictive models for IPL player price estimation using historical auction data sourced from Kaggle. The project aims to explore, preprocess,and analyze the dataset, addressing outliers and standardizing price features. Through machine learning techniques and feature selection, distinct models will be built for various player types, ensuring accuracy and reliability through robust validation methods. Additionally, a user-friendly web application will be created to facilitate real-time player price predictions based on player categories, offeringpractical utility for IPL auction predictions.

- To develop a IPL Auction Prediction Machine Learning Model using MLR, Random Forest, Gradient boosting Algorithm
- To predict the sold price of the player based upon the player type
- To Perform Exploratory Data Analysis to understand the distribution of data.
- To visualize the top 100 batsman and top 100 bowlers in ipl

CHAPTER 4
PROPOSED METHODOLOGY

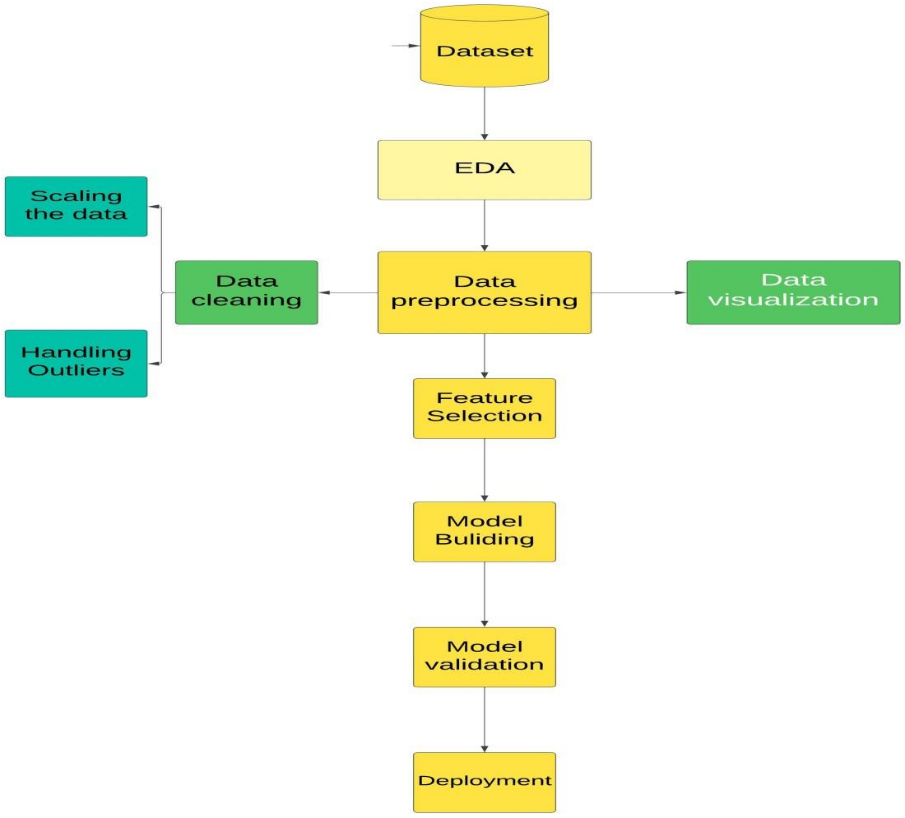


Fig 4.1 Methodology

The methodology employed for this project aimed to predict player prices in the IPL auction. It involved a systematic approach encompassing data preprocessing, exploratory analysis, feature selection, model building, validation, and application development. Each step was crucial in understanding, preparing, and analyzing the dataset to create predictive models for different player types.

.

CHAPTER 5
TOOLS AND TECHNIQUES

Jupyter: The newest web-based interactive development environment for code, data, and notebooks is called JupyterLab. Users can set up and organize workflows in data science, scientific computing, computational journalism, and machine learning thanks to its adaptable interface. Extensions are encouraged to increase and improve functionality in a modular design.

Scikit-learn: Scikit-learn is a popular machine learning library for Python that includes many algorithms for classification tasks, including deep learning algorithms such as multilayer perceptron (MLPs).

Libraries:

1.Matplotlib

- Plotting options offered by Matplotlib are extensive and include line, scatter, bar, and histogram plots, among other types. Matplotlib is a very flexible library.

- Plots can have different colors, labels, and styles customized by users thanks to its flexibility.

- Matplotlib is a well-liked option for data scientists and analysts working in interactive environments because of its smooth integration with Jupyter Notebooks. Plots in Jupyter Notebooks can be shown inline, enabling real-time data visualization and exploration.

2.Seaborn

- With a focus on statistical plotting, Seaborn excels at displaying complicated connections in datasets.

- It has pre-installed color schemes and themes to improve the plots' aesthetic appeal. Seaborn is an effective tool for quickly and easily producing meaningful statistical plots using the least amount of code because of its ease of use and integration with Pandas.

3.Plotly

- The Python package Plotly is flexible and can be used to make dynamic and interactive visuals. Zooming, panning, and hovering over data points facilitates dataexploration and analysis. Users can create interactive plots, charts, and dashboards with this tool.

- Line charts, bar charts, scatter plots, heatmaps, and three-dimensional charts are just a few of the many chart types that Plotly supports. Due to this adaptability, it can be used for a wide range of data visualization applications, from straightforward exploratory plots to intricate multi-dimensional visualizations.

CHAPTER 6
IMPLEMENTATION

6.1. ABOUT THE DATASETS:

6.1.1. Auction Dataset:

- Source of the Dataset : kaggle
- No. of Observations : 130
- Column/Feature Details : 26 columns
- Details about the columns :
- Discrete - T-runs,T-wkts,O-Runs,O-sr,O-wkts,e.tc
- Categorical - Playing Role

6.1.2.Top 100 Batsmen:

- Source of the Dataset : kaggle
- No. of Observations : 100
- Column/Feature Details : 14 columns
- Details about the columns :
- Discrete - matches, inn, no, runs, hs, 100, 50, 4s, 6s.
- continuous - avg,sr

6.1.3. Top 100 Bowlers:

- Source of the Dataset : kaggle
- No. of Observations : 100
- Column/Feature Details : 14 columns
- Details about the columns :
- Discrete - matches, inn, overs, runs, wkts, 4w, 5w.
- continuous - avg,sr, econ,bbi

6.2. EXPLORATORY DATA ANALYSIS

In the field of IPL auction prediction models, exploratory data analysis (EDA) is a fundamental tenet that is essential to model development. In this particular context, exploratory data analysis (EDA) focuses on analyzing past auction data to identify trends and derive significant insights that are essential to the predictive modeling procedure. When it comes to forecasting player values in IPL auctions, EDA entails a thorough investigation of several aspects, such as player demographics, historical performance metrics, club tactics, and auction dynamics.

6.3. FEATURE EXTRACTION

Key insights from the IPL auction dataset have been extracted primarily through theuse of heatmap visualization for feature extraction. This method has allowed for the successful identification and isolation of critical patterns and discriminative characteristics that are necessary for predicting player prices.

The utilization of heatmap visualization is an effective method for identifying the connections and comparative importance of various elements in the dataset. This research helped identify key factors that have a substantial impact on player valuations in the context of IPL auction prediction. These factors include player performance measures, prior auction valuations, team preferences, and market dynamics.

6.4. PRE-PROCESSING

Preprocessing is critical to improving the IPL auction prediction model, and resilient scaling is a key strategy that is similar to standard scaling for dataset normalization. Unlike normal scaling, robust scaling provides robustness against outliers and various data scales found in auction data, hence guaranteeing model stability and dependability.

A key technique in model creation is the train-test split methodology, which divides the dataset into distinct training and testing groups. The testing set assesses the model's performance, prevents overfitting, and determines whether the model can successfully generalize to fresh auction data, whereas the training set teaches the model.

Robust scaling uses robust statistical variables like the median and interquartile range for normalization, as opposed to standard scaling's use of the mean and standard deviation. This method maintains the integrity of the data, guarantees normalization while reducing the impact of outliers, and strengthens the model's capacity to adjust to changing auction dynamics.

6.5. MODELS

After having performed data preprocessing, we apply Regression models, namely

Multiple Linear Regression

- One of the fundamental techniques in machine learning is multiple linear regression, which is typically used for regression problems in predictive modeling. In contrast to ensemble techniques such as Random Forest, Multiple Linear Regression functions based on a unique approach derived from mathematical formulas.
- In order to forecast the value of the dependent variable based on the provided predictors, multiple linear regression essentially creates a linear relationship between a number of independent factors and a dependent variable. This approach creates a linear equation that estimates the value of the target variable by using coefficients to determine how important each predictor variable is.
-

Random Forest

- Random Forest is an ensemble learning algorithm that is commonly used for classification and regression tasks in machine learning.
- Random Forest algorithm is based on probability.
- To make a prediction on new data, the random forest algorithm first runseach data point through each of the individual decision trees in the forest, and then takes the majority vote of the predicted class (in classification tasks)or the average prediction (in regression tasks).

Gradient boosting

- Within the field of machine learning, gradient boosting is another well-liked ensemble learning technique that is frequently applied to tasks involving both regression and classification. Gradient Boosting, in contrast to Multiple Linear Regression, is an iterative ensemble technique that sequentially combines several weak learners, typically decision trees, to create a strongpredictive model.
- Gradient Boosting, which functions differently from linear regression, trains decision trees one after the other in order to rectify the mistakes made by the earlier trees. The goal of each successive tree is to reduce the residuals, or errors, left by the preceding tree, thus improving the prediction power of the model.

6.6. EVALUATION

Mean Square Error

- The Mean Squared Error (MSE) is a commonly used metric to evaluate the performance of a regression model. It quantifies the average squared differencebetween the actual (observed) values and the predicted values produced by themodel.
- An understanding of the model's accuracy may be gained from the MSE value; a lower MSE suggests that the model performed better in minimizing prediction errors and that its forecasts were more in line with actual values. On the other hand,a higher MSE suggests that the model is less accurate in its predictions because it deviates further from the actual values.

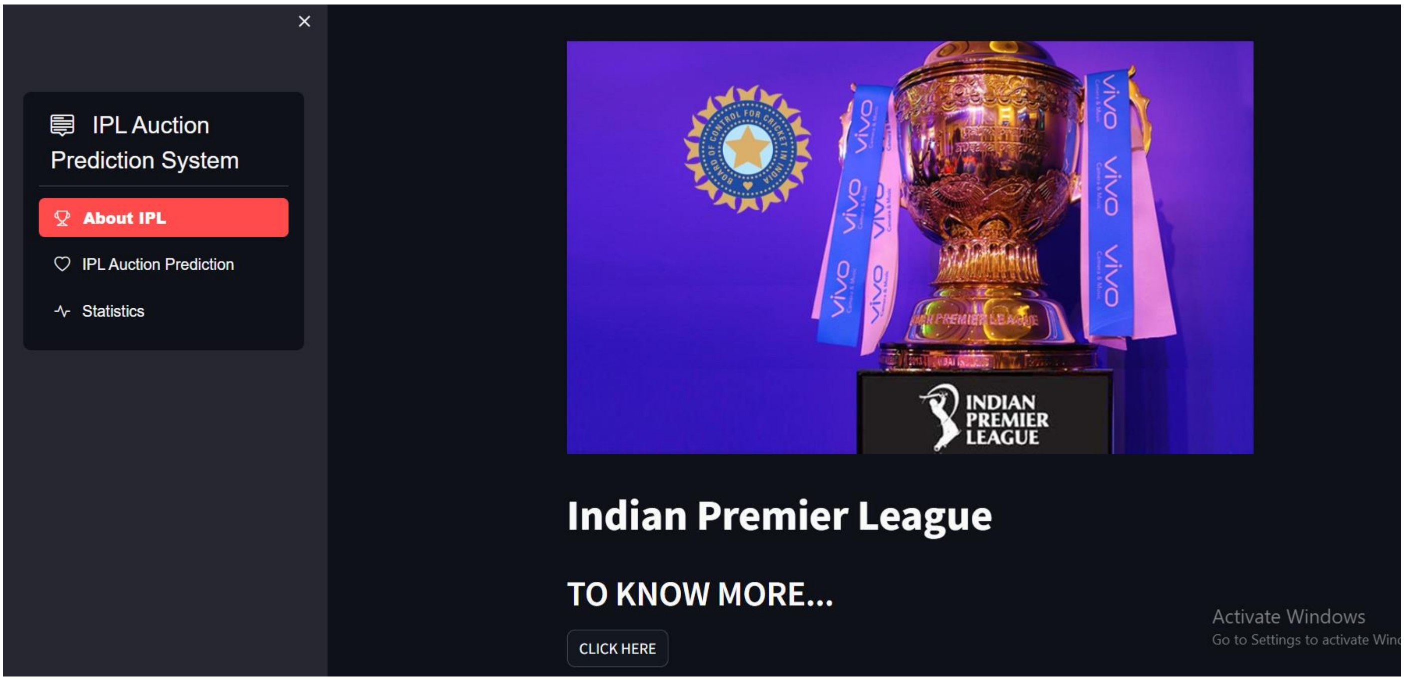
CHAPTER 7

RESULTS AND DISCUSSIONS

The evaluation of the IPL auction prediction models demonstrates varying performance metrics across different algorithms, highlighting their efficacy in estimating player prices. The Multi-Linear Regression (MLP) model achieved an Mean Absolute Error(MSE) of 0.878, 0.461 and 0.271 for all rounders,batsmen, and bowlers respectively. Meanwhile, the Random Forest (RF) model exhibited an MSE of 0.905, 0.648 and 0.288 for all rounders,batsmen, and bowlers respectively. Notably, the Gradient Boosting (GB) model has an MSE of 0.9241,1.173 and 0.443 which are larger compared to other models . These results emphasize the capability of these machine learning models in estimating IPL playerprices. However, further improvements may be attainable with larger and morestandardized datasets, which could potentially enhance the predictive performance of these models for more precise auction price estimation.

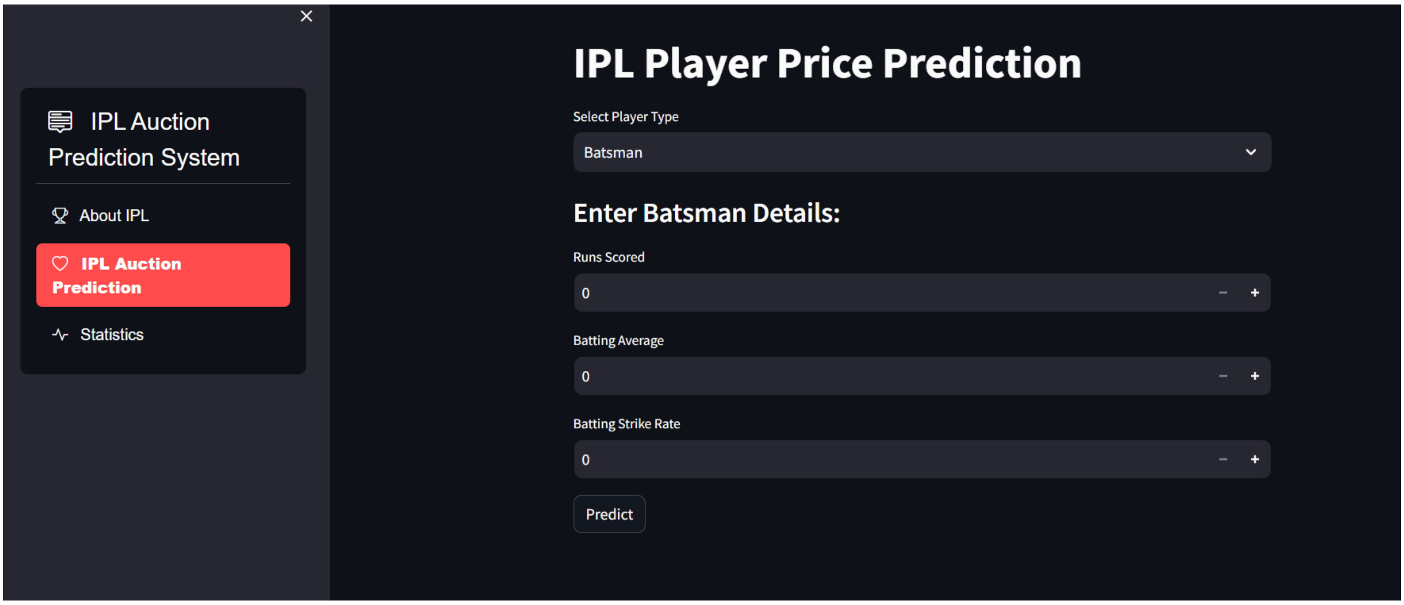
Models	Mean Square Error(MSE)		
	all-rounder	batsmen	bowlers
Multiple linear Regression	0.796	0.471	0.218
Random Forest Regressor	0.970	0.646	0.288
Gradient Boosting	0.878	1.173	0.444

Table 7.1 Results of Machine learning models



Indian Premier League

TO KNOW MORE...



IPL Player Price Prediction

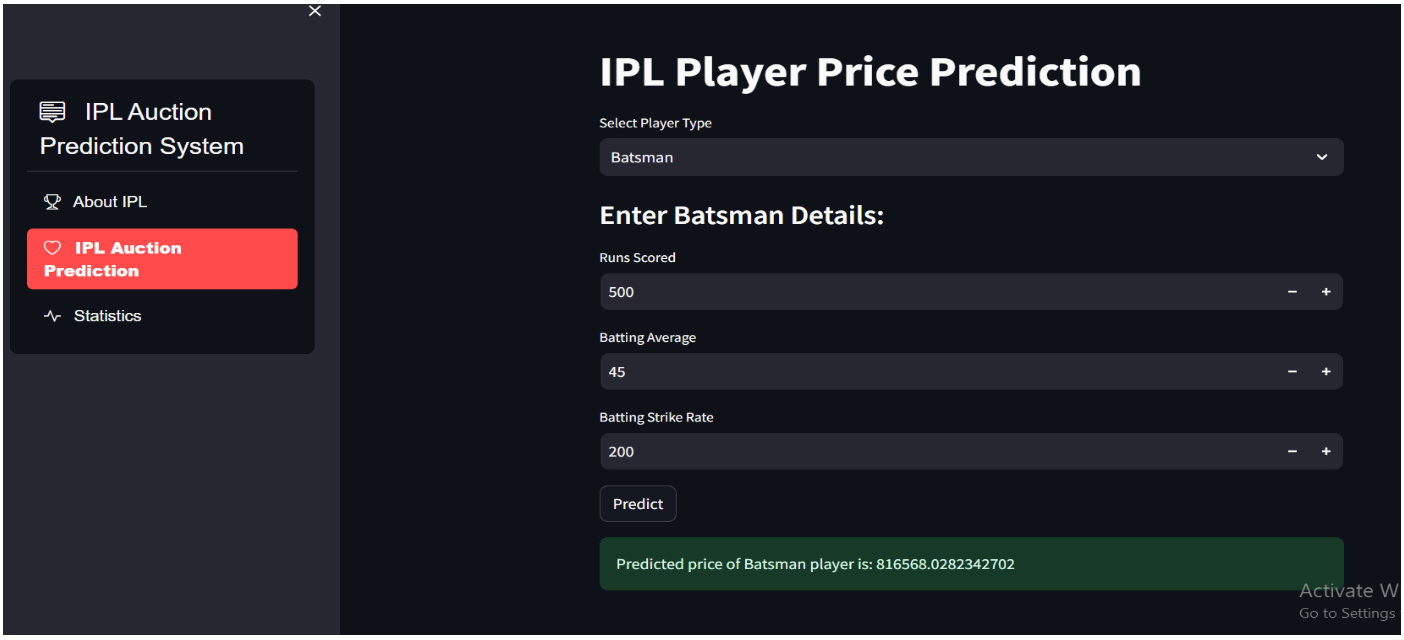
Select Player Type: Batsman

Enter Batsman Details:

Runs Scored: 0

Batting Average: 0

Batting Strike Rate: 0



IPL Player Price Prediction

Select Player Type: Batsman

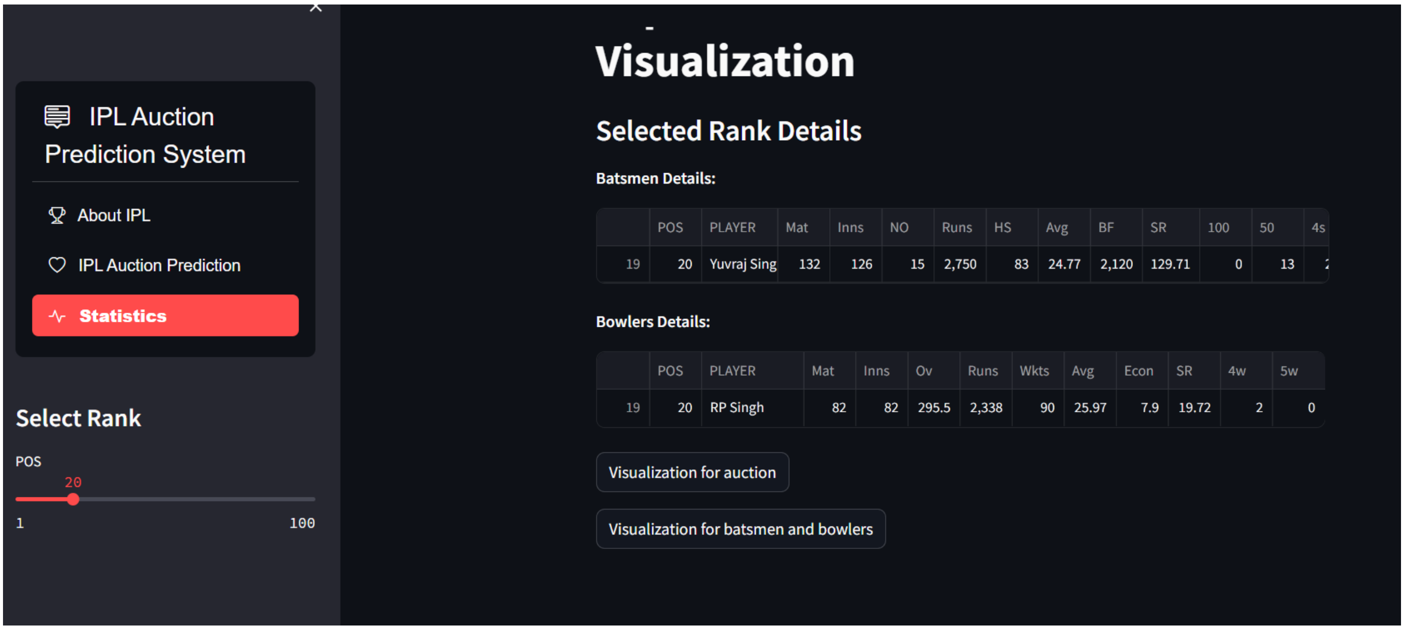
Enter Batsman Details:

Runs Scored: 500

Batting Average: 45

Batting Strike Rate: 200

Predicted price of Batsman player is: 816568.0282342702



IPL Auction Prediction System

- About IPL
- IPL Auction Prediction
- Statistics

Select Rank

POS

20

1 100

Visualization

Selected Rank Details

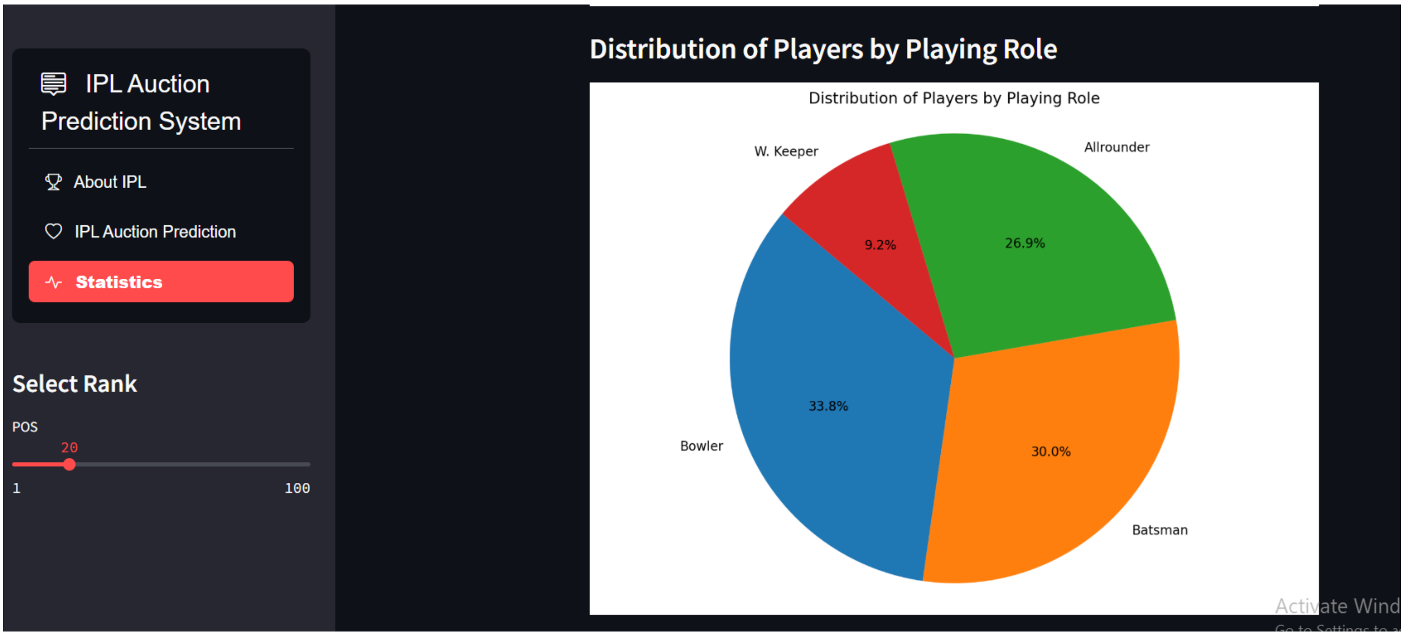
Batsmen Details:

	POS	PLAYER	Mat	Inns	NO	Runs	HS	Avg	BF	SR	100	50	4s
19	20	Yuvraj Sing	132	126	15	2,750	83	24.77	2,120	129.71	0	13	2

Bowlers Details:

	POS	PLAYER	Mat	Inns	Ov	Runs	Wkts	Avg	Econ	SR	4w	5w
19	20	RP Singh	82	82	295.5	2,338	90	25.97	7.9	19.72	2	0

Visualization for auction
Visualization for batsmen and bowlers



IPL Auction Prediction System

- About IPL
- IPL Auction Prediction
- Statistics

Select Rank

POS

20

1 100

Distribution of Players by Playing Role

Distribution of Players by Playing Role
W. Keeper
Allrounder
9.2%
26.9%
33.8%
Bowler
30.0%
Batsman

CHAPTER 8

CONCLUSION

- In conclusion, the IPL auction prediction models developed using machine learning techniques showcase promising capabilities in estimating player prices. The models, particularly the Multi-Linear Regression (MLR), Random Forest (RF), and Gradient Boosting (GB), demonstrate Mean Absolute Error in forecasting player prices.

- This study underscores the potential of machine learning algorithms in predicting auction prices based on historical data, offering insights into player valuation dynamics crucial for team strategies.
- However, while these models exhibit competence in estimating prices, furtherrefinement is warranted. Larger and standardized datasets could potentially enhancemodel performance, fostering more precise and reliable player price predictions. Additionally, ongoing research could delve deeper into feature engineering, model optimization, and real-time market dynamics to fortify the models' practical applicability in the dynamic context of IPL auctions.

CHAPTER 9

FUTURE ENHANCEMENT

- To further enhance the IPL auction prediction model, future research can explore incorporating advanced machine learning algorithms, leveraging real-time data during auctions, and integrating qualitative factors such as player consistency and adaptability
- To predict the price the based on the type of bowler (fast, off spin, leg spin)

CHAPTER 10

APPENDICES

Full code :

#Importing the Dependencies

```
import numpy as np

import pndas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline
```

# Loading Datasets

```
auction = pd.read_csv("C:/Users/23MSP3017/Downloads/IPL
IMB381IPL2013 (1).csv")

batsmen =
pd.read_excel("C:/Users/23MSP3017/Downloads/Top_100_batsman
(1).xlsx")

bowlers =
pd.read_excel("C:/Users/23MSP3017/Downloads/Top_100_bowlers.xlsx")
```

# Checking Null Values

```
auction.isnull().sum()
batsmen.isnull().sum()
bowlers.isnull().sum()
```

# Scaling

```
from  sklearn.preprocessing  import  RobustScaler
scaler = RobustScaler()
columns_to_scale = ['BASE PRICE', 'SOLD PRICE']

auction[columns_to_scale] =
scaler.fit_transform(auction[columns_to_scale])

print(auction)
```

# linear regression for allrounder

```
from sklearn.model_selection import train_test_splitfrom
sklearn.linear_model import LinearRegression from
sklearn.metrics import mean_squared_error
from sklearn.preprocessing import RobustScaler



all_rounder_data = cleaned_data[cleaned_data['PLAYING ROLE'] ==
'Allrounder']


#  Define  features  and  target  for  batsman  model
all_rounder_features = ['RUNS-S', 'AVE',
'SR-B','RUNS-C','WKTS','ECON','SR-BL','AVE-BL']

all_rounder_target = 'SOLD PRICE'
```

```
X_train_all_rounder, X_test_all_rounder, y_train_all_rounder,
y_test_all_rounder = train_test_split(
    all_rounder_data[all_rounder_features],
all_rounder_data[all_rounder_target],
test_size=0.2,train_size=0.8, random_state=42
)



# Initialize and train the Multiple Linear Regression model
lr_model = LinearRegression() lr_model.fit(X_train_all_rounder,
y_train_all_rounder)


# Make predictions on the test set

predictions_lr = lr_model.predict(X_test_all_rounder)



# Calculate Mean Squared Error for Linear Regression model

mse_lr_allrounder = mean_squared_error(y_test_all_rounder,
predictions_lr)
print(f"Multiple Linear Regression Model Mean Squared Error:
{mse_lr_allrounder}")





new_player_features = [300, 45, 130,250,30,7,20,24]  # Replace
these values with the new player's data



# Reshape the input data to match the model's requirements

new_player_features_reshaped =
np.array(new_player_features).reshape(1, -1)



# Predict the 'SOLD PRICE' for the new player

predicted_price_scaled =
lr_model.predict(new_player_features_reshaped)
```

```
# Inverse transform the predicted 'SOLD PRICE' to the originalscale

predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]



print(f"Predicted 'SOLD PRICE' for allrounder:

{predicted_price_original}")
linear regression for batsmen
from sklearn.model_selection import train_test_splitfrom
sklearn.linear_model import LinearRegression from
sklearn.metrics import mean_squared_error


batsman_data = cleaned_data[cleaned_data['PLAYING ROLE'] ==
'Batsman']



# Define features and target for batsman model
batsman_features = ['RUNS-S', 'AVE', 'SR-B']
batsman_target = 'SOLD PRICE'


# Split data for batsmen

X_train_batsman, X_test_batsman, y_train_batsman, y_test_batsman =
train_test_split(

    batsman_data[batsman_features], batsman_data[batsman_target],
test_size=0.2, random_state=42

)



# Initialize and train the Multiple Linear Regression model
lr_model = LinearRegression()
lr_model.fit(X_train_batsman, y_train_batsman)



# Make predictions on the test set
```

```
predictions_lr = lr_model.predict(X_test_batsman)



# Calculate Mean Squared Error for Linear Regression model

mse_lr_batsman = mean_squared_error(y_test_batsman,
predictions_lr)
print(f"Multiple Linear Regression Model Mean Squared Error:
{mse_lr_batsman}")



# Example data for a new 'Batsman' player

new_player_features = [300, 45, 130]  # Replace these values withthe
new player's data



# Reshape the input data to match the model's requirements

new_player_features_reshaped =
np.array(new_player_features).reshape(1, -1)



# Predict the 'SOLD PRICE' for the new player

predicted_price_scaled =
lr_model.predict(new_player_features_reshaped)



# Inverse transform the predicted 'SOLD PRICE' to the originalscale
predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]


# Display the predicted price for the new player in both scales
print(f"Predicted 'SOLD PRICE' for Batsman:
{predicted_price_original}")





bowler_data = cleaned_data[cleaned_data['PLAYING ROLE'] ==
'Bowler']
```

```
# Define features and target for batsman model
bowler_features = ['RUNS-C', 'AVE-BL', 'SR-BL','WKTS']
bowler_target = 'SOLD PRICE'


# Split data for batsmen

X_train_bowler, X_test_bowler, y_train_bowler, y_test_bowler =
train_test_split(

    bowler_data[bowler_features], bowler_data[bowler_target],
test_size=0.2, random_state=42

)



# Initialize and train the Multiple Linear Regression model
lr_model = LinearRegression()
lr_model.fit(X_train_bowler, y_train_bowler)



# Make predictions on the test set predictions_lr =
lr_model.predict(X_test_bowler)

# Calculate Mean Squared Error for Linear Regression model
mse_lr_bowler = mean_squared_error(y_test_bowler, predictions_lr)
print(f"Multiple Linear Regression Model Mean Squared Error:
{mse_lr_bowler}")



new_player_features = [500,25,22,21]  # Replace these values withthe
new player's data



# Reshape the input data to match the model's requirements

new_player_features_reshaped =
np.array(new_player_features).reshape(1, -1)
```

```
# Predict the 'SOLD PRICE' for the new player

predicted_price_scaled =
lr_model.predict(new_player_features_reshaped)



# Inverse transform the predicted 'SOLD PRICE' to the originalscale

predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]


# Display the predicted price for the new player in both scales
print(f"Predicted 'SOLD PRICE' for Bowler :

{predicted_price_original}")



import pandas as pd

from sklearn.model_selection import train_test_splitfrom
sklearn.ensemble import RandomForestRegressor from
sklearn.metrics import mean_squared_error


batsman_data = cleaned_data[cleaned_data['PLAYING ROLE'] ==
'Batsman']



# Define features and target for batsman model
batsman_features = ['RUNS-S', 'AVE', 'SR-B']
batsman_target = 'SOLD PRICE'


# Split data for batsmen

X_train_batsman, X_test_batsman, y_train_batsman, y_test_batsman =
train_test_split(

    batsman_data[batsman_features], batsman_data[batsman_target],
test_size=0.2, random_state=42

)
```

```
# Train a model specific to batsmen

batsman_model = RandomForestRegressor(n_estimators=100,
random_state=42)

batsman_model.fit(X_train_batsman, y_train_batsman)



# Make predictions on the test set for batsmen
predictions_batsman = batsman_model.predict(X_test_batsman)


# Calculate Mean Squared Error for batsman model

mse_rf_batsman = mean_squared_error(y_test_batsman,
predictions_batsman)

print(f"Batsman Model Mean Squared Error: {mse_rf_batsman}")



new_player_features = [300, 45, 130]  # Replace these values withthe
new player's data



# Reshape the input data to match the model's requirements

new_player_features_reshaped =
np.array(new_player_features).reshape(1, -1)



# Predict the 'SOLD PRICE' for the new player

predicted_price_scaled =
batsman_model.predict(new_player_features_reshaped)



# Inverse transform the predicted 'SOLD PRICE' to the originalscale

predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]



# Display the predicted price for the new player in both scales
```

```
print(f"Predicted 'SOLD PRICE' for Batsman:
{predicted_price_original}")



bowler_data = cleaned_data[cleaned_data['PLAYING ROLE'] ==
'Bowler']



# Define features and target for batsman model
bowler_features = ['RUNS-C', 'AVE-BL', 'SR-BL','WKTS']
bowler_target = 'SOLD PRICE'


# Split data for batsmen

X_train_bowler, X_test_bowler, y_train_bowler, y_test_bowler =
train_test_split(

    bowler_data[bowler_features], bowler_data[bowler_target],
test_size=0.2, random_state=42

)

# Train a model specific to batsmen

bowler_model = RandomForestRegressor(n_estimators=100,
random_state=42)

bowler_model.fit(X_train_bowler, y_train_bowler)



# Make predictions on the test set for batsmen
predictions_bowler = bowler_model.predict(X_test_bowler)


# Calculate Mean Squared Error for batsman model

mse_rf_bowler = mean_squared_error(y_test_bowler,
predictions_bowler)

print(f"Batsman Model Mean Squared Error: {mse_rf_bowler}")



new_player_features = [500,25,22,21]  # Replace these values withthe
new player's data
```

```
# Reshape the input data to match the model's requirements

new_player_features_reshaped =
np.array(new_player_features).reshape(1, -1)



# Predict the 'SOLD PRICE' for the new player

predicted_price_scaled =
bowler_model.predict(new_player_features_reshaped)



# Inverse transform the predicted 'SOLD PRICE' to the originalscale

predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]


# Display the predicted price for the new player in both scales
print(f"Predicted 'SOLD PRICE' for Bowler :
{predicted_price_original}")

# Filter data for all-rounders

all_rounder_data = cleaned_data[cleaned_data['PLAYING ROLE'] ==
'Allrounder']



# Define features and target for all-rounder model
all_rounder_features = ['RUNS-S', 'AVE', 'SR-B','ECON',
                        'RUNS-C', 'AVE-BL', 'SR-BL','WKTS']

all_rounder_target = 'SOLD PRICE'



# Split data for all-rounders

X_train_all_rounder, X_test_all_rounder, y_train_all_rounder,
y_test_all_rounder = train_test_split(

    all_rounder_data[all_rounder_features],
all_rounder_data[all_rounder_target],
test_size=0.2,train_size=0.8, random_state=42
)



# Train a model specific to all-rounders

all_rounder_model = RandomForestRegressor(n_estimators=100,
random_state=42)

all_rounder_model.fit(X_train_all_rounder, y_train_all_rounder)



# Make predictions on the test set for all-rounders

predictions_all_rounder =
all_rounder_model.predict(X_test_all_rounder)



# Calculate Mean Squared Error for all-rounder model

mse_rf_all_rounder = mean_squared_error(y_test_all_rounder,
predictions_all_rounder)

print(f"All-Rounder Model Mean Squared Error:
{mse_rf_all_rounder}")



new_player_features = [300, 45, 130,250,30,7,20,24]  # Replace
these values with the new player's data



# Reshape the input data to match the model's requirements

new_player_features_reshaped =
np.array(new_player_features).reshape(1, -1)



# Predict the 'SOLD PRICE' for the new player

predicted_price_scaled =
all_rounder_model.predict(new_player_features_reshaped)



# Inverse transform the predicted 'SOLD PRICE' to the originalscale

predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]
```

```
print(f"Predicted 'SOLD PRICE' for allrounder:
{predicted_price_original}")

from sklearn.ensemble import GradientBoostingRegressor



batsman_data = cleaned_data[cleaned_data['PLAYING ROLE'] ==
'Batsman']



# Define features and target for batsman model
batsman_features = ['RUNS-S', 'AVE', 'SR-B']
batsman_target = 'SOLD PRICE'


X_train_batsman, X_test_batsman, y_train_batsman, y_test_batsman =
train_test_split(

    batsman_data[batsman_features], batsman_data[batsman_target],
test_size=0.2, random_state=42

)



gb_model = GradientBoostingRegressor(n_estimators=100,
learning_rate=0.1, random_state=42)

gb_model.fit(X_train_batsman, y_train_batsman)



# Make predictions on the test set

predictions_gb = gb_model.predict(X_test_batsman)



# Calculate Mean Squared Error for Gradient Boosting model

mse_gb_batsman = mean_squared_error(y_test_batsman,
predictions_gb)

print(f"Gradient Boosting Regressor Model Mean Squared Error:
{mse_gb_batsman}")
```

```
new_player_features = [300, 45, 130]  # Replace these values withthe
new player's data


# Reshape the input data to match the model's requirements

new_player_features_reshaped =
np.array(new_player_features).reshape(1, -1)


# Predict the 'SOLD PRICE' for the new player

predicted_price_scaled =
gb_model.predict(new_player_features_reshaped)


# Inverse transform the predicted 'SOLD PRICE' to the originalscale

predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]


# Display the predicted price for the new player in both scales
print(f"Predicted 'SOLD PRICE' for Batsman:
{predicted_price_original}")

# Filter data for all-rounders

all_rounder_data = cleaned_data[cleaned_data['PLAYING ROLE'] ==
'Allrounder']


# Define features and target for all-rounder model
all_rounder_features = ['RUNS-S', 'AVE', 'SR-B','ECON',
                        'RUNS-C', 'AVE-BL', 'SR-BL','WKTS']

all_rounder_target = 'SOLD PRICE'


# Split data for all-rounders

X_train_all_rounder, X_test_all_rounder, y_train_all_rounder,
y_test_all_rounder = train_test_split(
    all_rounder_data[all_rounder_features],
all_rounder_data[all_rounder_target],
test_size=0.2,train_size=0.8, random_state=42

)



gb_model = GradientBoostingRegressor(n_estimators=100,
learning_rate=0.1, random_state=42)

gb_model.fit(X_train_all_rounder, y_train_all_rounder)



# Make predictions on the test set

predictions_gb = gb_model.predict(X_test_all_rounder)



# Calculate Mean Squared Error for Gradient Boosting model

mse_gb_allrounder = mean_squared_error(y_test_all_rounder,
predictions_gb)

print(f"Gradient Boosting Regressor Model Mean Squared Error:

{mse_gb_allrounder}")



new_player_features = [300, 45, 130,250,30,7,20,24]  # Replace
these values with the new player's data



# Reshape the input data to match the model's requirements

new_player_features_reshaped =
np.array(new_player_features).reshape(1, -1)



# Predict the 'SOLD PRICE' for the new player

predicted_price_scaled =
gb_model.predict(new_player_features_reshaped)



# Inverse transform the predicted 'SOLD PRICE' to the originalscale
```

```
predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]



print(f"Predicted 'SOLD PRICE' for allrounder:
{predicted_price_original}")

bowler_data = cleaned_data[cleaned_data['PLAYING ROLE'] ==
'Bowler']



# Define features and target for batsman model
bowler_features = ['RUNS-C', 'AVE-BL', 'SR-BL','WKTS']
bowler_target = 'SOLD PRICE'


# Split data for batsmen

X_train_bowler, X_test_bowler, y_train_bowler, y_test_bowler =
train_test_split(

    bowler_data[bowler_features], bowler_data[bowler_target],
test_size=0.2, random_state=42

)
```

00

```
gb_model = GradientBoostingRegressor(n_estimators=100,
learning_rate=0.1, random_state=42)

gb_model.fit(X_train_bowler, y_train_bowler)





predictions_gb = gb_model.predict(X_test_bowler)


mse_gb_bowler = mean_squared_error(y_test_bowler, predictions_gb)
print(f"Gradient Boosting Regressor Model Mean Squared Error:
{mse_gb_bowler}")



new_player_features = [500,25,22,21]  # Replace these values withthe
new player's data
```

```
# Reshape the input data to match the model's requirements

new_player_features_reshaped =
np.array(new_player_features).reshape(1, -1)



# Predict the 'SOLD PRICE' for the new player

predicted_price_scaled =
gb_model.predict(new_player_features_reshaped)



# Inverse transform the predicted 'SOLD PRICE' to the originalscale

predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]


# Display the predicted price for the new player in both scales
print(f"Predicted 'SOLD PRICE' for Bowler :

{predicted_price_original}")
import plotly.express as px




# Store the MSE values and their corresponding labels mse_values =
[mse_lr_batsman, mse_rf_batsman, mse_gb_batsman]labels = ['lr',
'rf', 'gb']


# Create a DataFrame for Plotly

data = {'Models': labels, 'MSE Values': mse_values}df =
pd.DataFrame(data)


# Create an interactive bar graph using Plotly

fig = px.bar(df, x='Models', y='MSE Values',color=['blue',
'green', 'orange'])
```

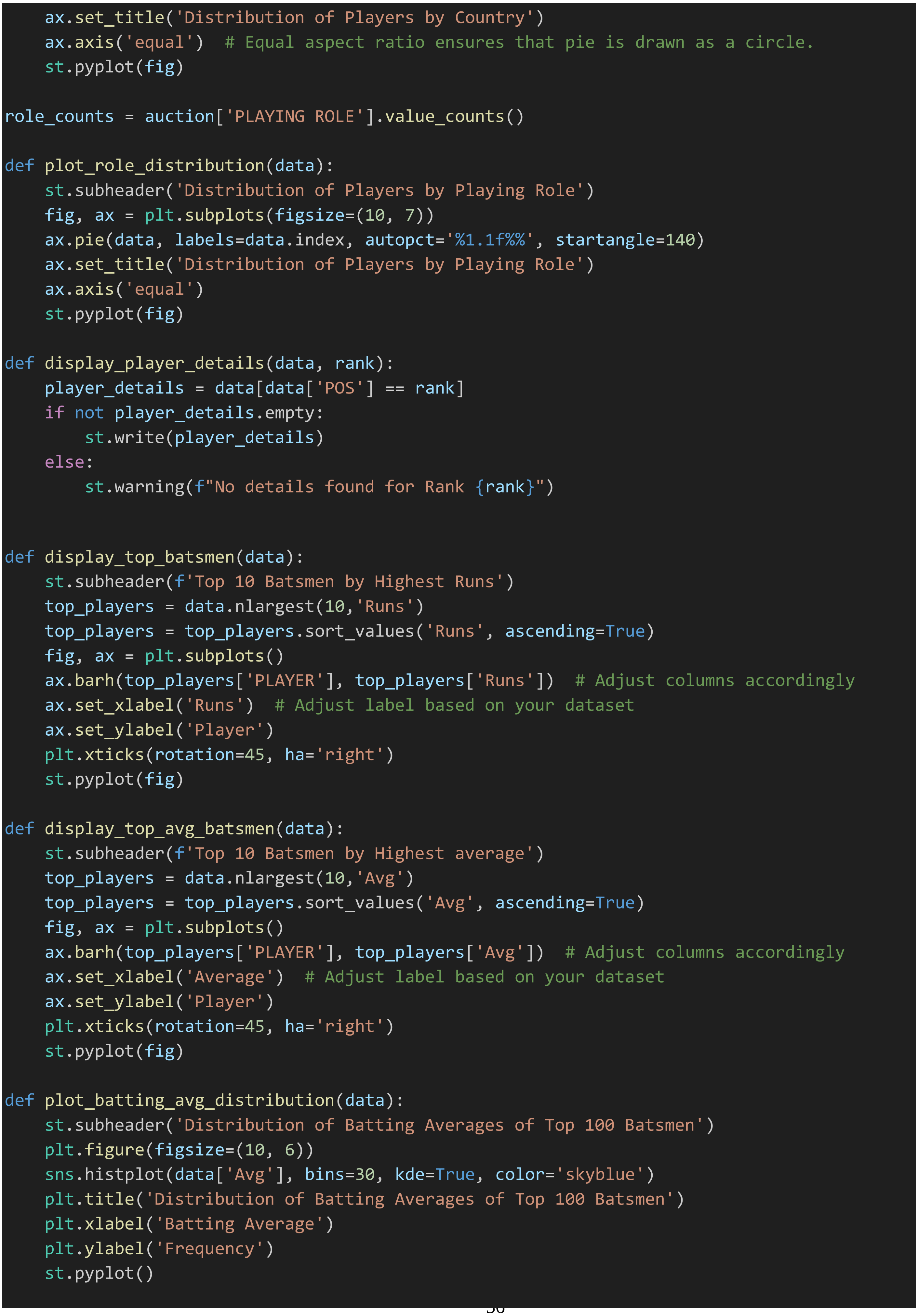
```
#fig.update_traces(texttemplate='%{text}', textposition='outside')
fig.update_layout(
    title='Mean Squared Error (MSE) for batsman',
    xaxis_title='Models',
    yaxis_title='MSE Values',
    hovermode='closest',
)



# Display the interactive graph
fig.show()


import plotly.express as px





# Store the MSE values and their corresponding labels
mse_values = [mse_lr_bowler, mse_rf_bowler, mse_gb_bowler]
labels = ['lr', 'rf', 'gb']


# Create a DataFrame for Plotly

data = {'Models': labels, 'MSE Values': mse_values}df =
pd.DataFrame(data)


# Create an interactive bar graph using Plotly

fig = px.bar(df, x='Models', y='MSE Values',color=['blue',
'green', 'orange'])

#fig.update_traces(texttemplate='%{text}', textposition='outside')
fig.update_layout(
    title='Mean Squared Error (MSE) for Bowlers',
    xaxis_title='Models',
      yaxis_title='MSE Values',
      hovermode='closest',
  )



  # Display the interactive graph
  fig.show()


  import plotly.express as px





  # Store the MSE values and their corresponding labels

  mse_values = [mse_lr_allrounder, mse_rf_all_rounder,
  mse_gb_allrounder]

  labels = ['lr', 'rf', 'gb']



  # Create a DataFrame for Plotly

  data = {'Models': labels, 'MSE Values': mse_values}df =
  pd.DataFrame(data)


  # Create an interactive bar graph using Plotly

  fig = px.bar(df, x='Models', y='MSE Values',color=['blue',
  'green', 'orange'])

  #fig.update_traces(texttemplate='%{text}', textposition='outside')
  fig.update_layout(
      title='Mean Squared Error (MSE) for all-rounders',
      xaxis_title='Models',
      yaxis_title='MSE Values',
      hovermode='closest',
  )

# Display the interactive graph

fig.show()
```

#Saving the trained model

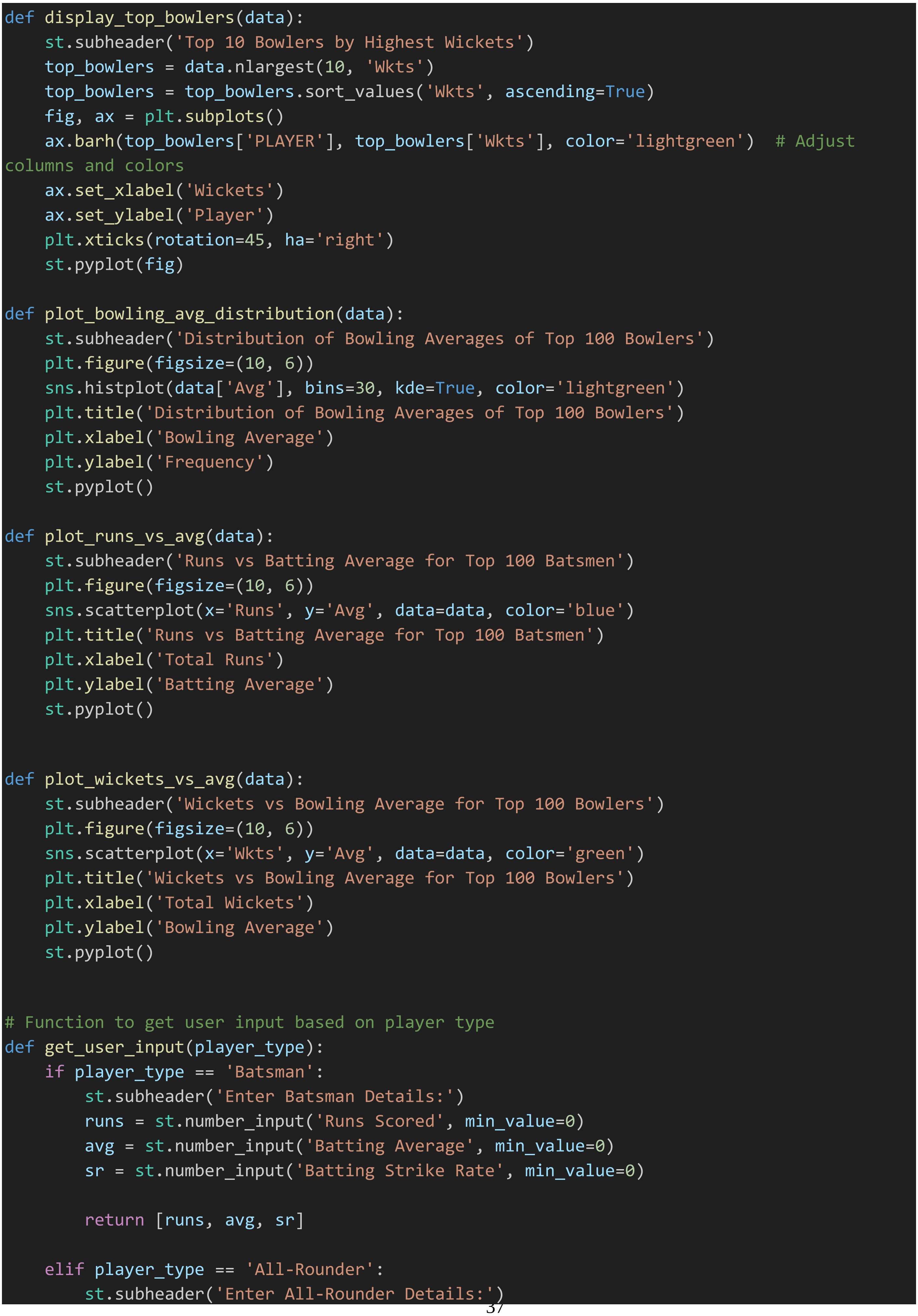
```
import pickle
filename = 'lr_model_allrounder.sav'
pickle.dump(lr_model_allrounder, open(filename, 'wb'))
# loading the saved model
loaded_model = pickle.load(open('lr_model_allrounder.sav', 'rb'))

import pickle
filename = 'lr_model_batsmen.sav'
pickle.dump(lr_model_batsmen, open(filename, 'wb'))
# loading the saved model
loaded_model = pickle.load(open('lr_model_batsmen.sav', 'rb'))

import pickle
filename = 'lr_model_bowler.sav'
pickle.dump(lr_model_bowler, open(filename, 'wb'))
# loading the saved model
loaded_model = pickle.load(open('lr_model_bowler.sav', 'rb'))
```

#FRONT-END

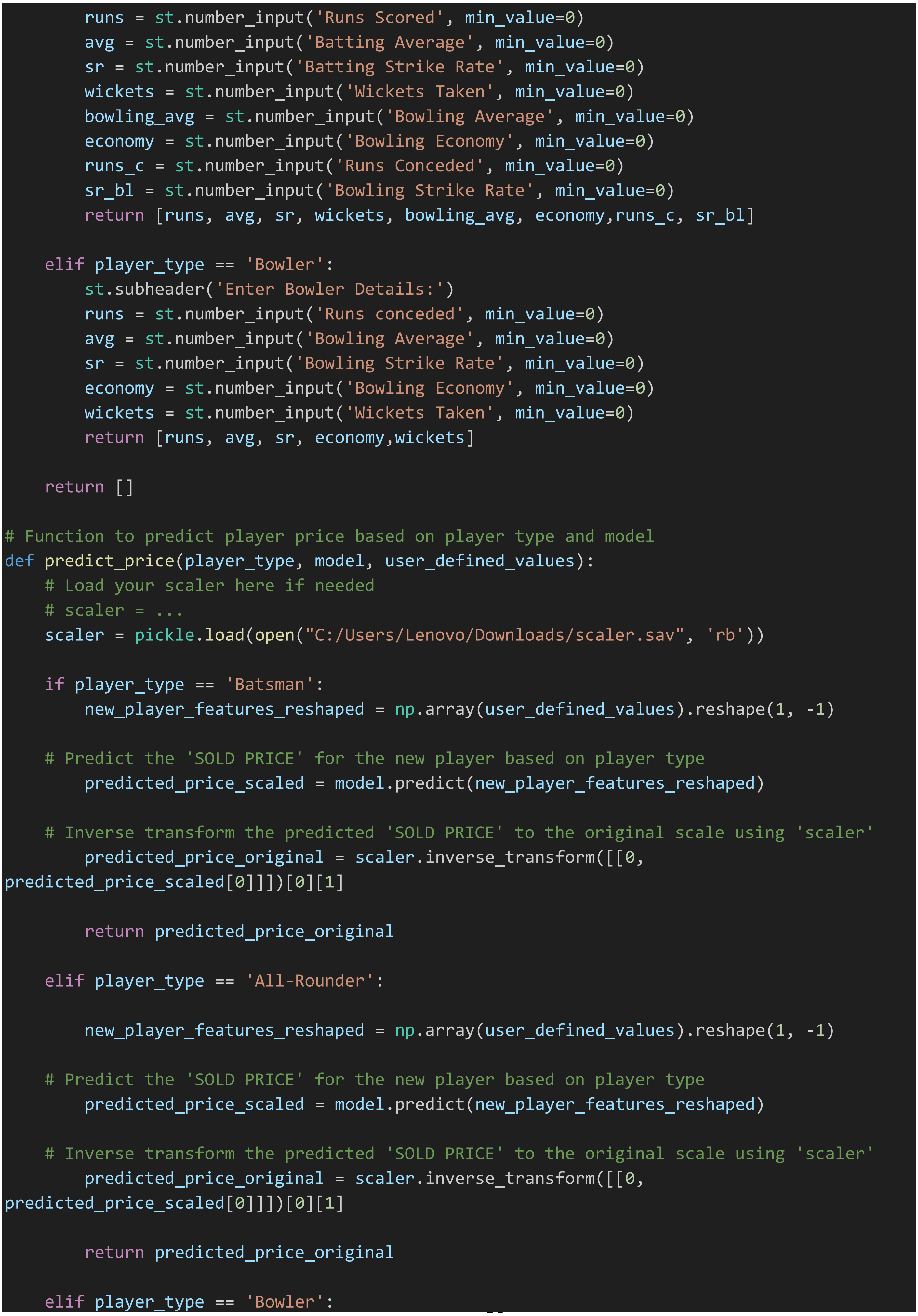
```
import streamlit as st
import pickle
import numpy as np
from streamlit_option_menu import option_menu
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

auction = pd.read_csv("C:/Users/Lenovo/Downloads/IPL IMB381IPL2013 (1).csv")
batsmen_data = pd.read_excel("C:/Users/Lenovo/Downloads/Top_100_batsman.xlsx")
bowlers_data = pd.read_excel("C:/Users/Lenovo/Downloads/Top_100_bowlers.xlsx")

auction_top10 = auction.nlargest(10, 'SOLD PRICE')

def plot_top10_buys(data):
    st.subheader('Top 10 Buys in IPL')
    plt.figure(figsize=(12, 8))
    plt.barh(data['PLAYER NAME'], data['SOLD PRICE'], color='skyblue')
    plt.xlabel('Sold Price')
    plt.title('Top 10 Buys in IPL')
    plt.gca().invert_yaxis()  # To display the highest price at the top
    st.pyplot()


country_counts = auction['COUNTRY'].value_counts()
def plot_country_distribution(data):
    st.subheader('Distribution of Players by Country')
    fig, ax = plt.subplots(figsize=(10, 7))
    ax.pie(data, labels=data.index, autopct='%1.1f%%', startangle=140)
```



```
    ax.set_title('Distribution of Players by Country')
    ax.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
    st.pyplot(fig)

role_counts = auction['PLAYING ROLE'].value_counts()

def plot_role_distribution(data):
    st.subheader('Distribution of Players by Playing Role')
    fig, ax = plt.subplots(figsize=(10, 7))
    ax.pie(data, labels=data.index, autopct='%1.1f%%', startangle=140)
    ax.set_title('Distribution of Players by Playing Role')
    ax.axis('equal')
    st.pyplot(fig)

def display_player_details(data, rank):
    player_details = data[data['POS'] == rank]
    if not player_details.empty:
        st.write(player_details)
    else:
        st.warning(f"No details found for Rank {rank}")


def display_top_batsmen(data):
    st.subheader(f'Top 10 Batsmen by Highest Runs')
    top_players = data.nlargest(10,'Runs')
    top_players = top_players.sort_values('Runs', ascending=True)
    fig, ax = plt.subplots()
    ax.barh(top_players['PLAYER'], top_players['Runs'])  # Adjust columns accordingly
    ax.set_xlabel('Runs')  # Adjust label based on your dataset
    ax.set_ylabel('Player')
    plt.xticks(rotation=45, ha='right')
    st.pyplot(fig)

def display_top_avg_batsmen(data):
    st.subheader(f'Top 10 Batsmen by Highest average')
    top_players = data.nlargest(10,'Avg')
    top_players = top_players.sort_values('Avg', ascending=True)
    fig, ax = plt.subplots()
    ax.barh(top_players['PLAYER'], top_players['Avg'])  # Adjust columns accordingly
    ax.set_xlabel('Average')  # Adjust label based on your dataset
    ax.set_ylabel('Player')
    plt.xticks(rotation=45, ha='right')
    st.pyplot(fig)

def plot_batting_avg_distribution(data):
    st.subheader('Distribution of Batting Averages of Top 100 Batsmen')
    plt.figure(figsize=(10, 6))
    sns.histplot(data['Avg'], bins=30, kde=True, color='skyblue')
    plt.title('Distribution of Batting Averages of Top 100 Batsmen')
    plt.xlabel('Batting Average')
    plt.ylabel('Frequency')
    st.pyplot()
```



```
def display_top_bowlers(data):
    st.subheader('Top 10 Bowlers by Highest Wickets')
    top_bowlers = data.nlargest(10, 'Wkts')
    top_bowlers = top_bowlers.sort_values('Wkts', ascending=True)
    fig, ax = plt.subplots()
    ax.barh(top_bowlers['PLAYER'], top_bowlers['Wkts'], color='lightgreen')  # Adjust
columns and colors
    ax.set_xlabel('Wickets')
    ax.set_ylabel('Player')
    plt.xticks(rotation=45, ha='right')
    st.pyplot(fig)

def plot_bowling_avg_distribution(data):
    st.subheader('Distribution of Bowling Averages of Top 100 Bowlers')
    plt.figure(figsize=(10, 6))
    sns.histplot(data['Avg'], bins=30, kde=True, color='lightgreen')
    plt.title('Distribution of Bowling Averages of Top 100 Bowlers')
    plt.xlabel('Bowling Average')
    plt.ylabel('Frequency')
    st.pyplot()

def plot_runs_vs_avg(data):
    st.subheader('Runs vs Batting Average for Top 100 Batsmen')
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='Runs', y='Avg', data=data, color='blue')
    plt.title('Runs vs Batting Average for Top 100 Batsmen')
    plt.xlabel('Total Runs')
    plt.ylabel('Batting Average')
    st.pyplot()


def plot_wickets_vs_avg(data):
    st.subheader('Wickets vs Bowling Average for Top 100 Bowlers')
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='Wkts', y='Avg', data=data, color='green')
    plt.title('Wickets vs Bowling Average for Top 100 Bowlers')
    plt.xlabel('Total Wickets')
    plt.ylabel('Bowling Average')
    st.pyplot()


# Function to get user input based on player type
def get_user_input(player_type):
    if player_type == 'Batsman':
        st.subheader('Enter Batsman Details:')
        runs = st.number_input('Runs Scored', min_value=0)
        avg = st.number_input('Batting Average', min_value=0)
        sr = st.number_input('Batting Strike Rate', min_value=0)

        return [runs, avg, sr]

    elif player_type == 'All-Rounder':
        st.subheader('Enter All-Rounder Details:')
```



```
        runs = st.number_input('Runs Scored', min_value=0)
        avg = st.number_input('Batting Average', min_value=0)
        sr = st.number_input('Batting Strike Rate', min_value=0)
        wickets = st.number_input('Wickets Taken', min_value=0)
        bowling_avg = st.number_input('Bowling Average', min_value=0)
        economy = st.number_input('Bowling Economy', min_value=0)
        runs_c = st.number_input('Runs Conceded', min_value=0)
        sr_bl = st.number_input('Bowling Strike Rate', min_value=0)
        return [runs, avg, sr, wickets, bowling_avg, economy,runs_c, sr_bl]

    elif player_type == 'Bowler':
        st.subheader('Enter Bowler Details:')
        runs = st.number_input('Runs conceded', min_value=0)
        avg = st.number_input('Bowling Average', min_value=0)
        sr = st.number_input('Bowling Strike Rate', min_value=0)
        economy = st.number_input('Bowling Economy', min_value=0)
        wickets = st.number_input('Wickets Taken', min_value=0)
        return [runs, avg, sr, economy,wickets]

    return []

# Function to predict player price based on player type and model
def predict_price(player_type, model, user_defined_values):
    # Load your scaler here if needed
    # scaler = ...
    scaler = pickle.load(open("C:/Users/Lenovo/Downloads/scaler.sav", 'rb'))

    if player_type == 'Batsman':
        new_player_features_reshaped = np.array(user_defined_values).reshape(1, -1)

    # Predict the 'SOLD PRICE' for the new player based on player type
        predicted_price_scaled = model.predict(new_player_features_reshaped)

    # Inverse transform the predicted 'SOLD PRICE' to the original scale using 'scaler'
        predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]

        return predicted_price_original

    elif player_type == 'All-Rounder':

        new_player_features_reshaped = np.array(user_defined_values).reshape(1, -1)

    # Predict the 'SOLD PRICE' for the new player based on player type
        predicted_price_scaled = model.predict(new_player_features_reshaped)

    # Inverse transform the predicted 'SOLD PRICE' to the original scale using 'scaler'
        predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]

        return predicted_price_original

    elif player_type == 'Bowler':
        new_player_features_reshaped = np.array(user_defined_values).reshape(1, -1)

    # Predict the 'SOLD PRICE' for the new player based on player type
        predicted_price_scaled = model.predict(new_player_features_reshaped)

    # Inverse transform the predicted 'SOLD PRICE' to the original scale using 'scaler'
        predicted_price_original = scaler.inverse_transform([[0,
predicted_price_scaled[0]]])[0][1]

        return predicted_price_original



    return "Invalid player type selected"


def main():
    st.set_page_config(page_title='IPL Player Prediction',
page_icon=":cricket_bat_and_ball:")



    st.set_option('deprecation.showPyplotGlobalUse', False)

    with st.sidebar:
        selected = option_menu('IPL Auction Prediction System',['About IPL','IPL Auction
Prediction','Statistics'],icons=['trophy','heart','activity'],
 default_index=0)

    if (selected == 'About IPL'):

        st.image('images/ipl-logo.jpg')

 # page title
        st.title('Indian Premier League')
        st.header('TO KNOW MORE...')
        re=st.button('CLICK HERE')

        st.write(":cricket_bat_and_ball:")
        if re:
            st.text("The Indian Premier League (IPL) is a men's Twenty20 (T20) cricket
league")
            st.text('The IPL is the most-popular cricket league in the world')
            st.text('It was ranked sixth by average attendance among all sports leagues.')
            st.text('The brand value of the league in 2022 was ₹90,038 crore (US$11
billion).')
            st.text('Its 2023 final was the most streamed live event on internet with 3.2
crore viewers')

    if (selected == 'IPL Auction Prediction'):
        st.image('images/auction.jpg', caption='ipl auction')
        st.title('IPL Player Price Prediction')

        # Dropdown to select player type
        player_types = ['Batsman', 'All-Rounder','Bowler']  # Add other player types if needed
        selected_player_type = st.selectbox('Select Player Type', player_types)

        # Get user input based on selected player type
        user_input = get_user_input(selected_player_type)

        button_color = "#008CBA"  # Blue color (you can change this)

        if st.button('Predict'):
            batsman_model = pickle.load(open("C:/Users/Lenovo/Downloads/lr_model_batsmen.sav", 'rb'))
            allrounder_model = pickle.load(open("C:/Users/Lenovo/Downloads/lr_model_allrounder.sav", 'rb'))
            bowler_model = pickle.load(open("C:/Users/Lenovo/Downloads/lr_model_bowler.sav", 'rb'))

            # Perform prediction based on the selected player type and user input
            if selected_player_type == 'Batsman':
                predicted_price = predict_price(selected_player_type, batsman_model, user_input)
            elif selected_player_type == 'All-Rounder':
                predicted_price = predict_price(selected_player_type, allrounder_model, user_input)
            elif selected_player_type == 'Bowler':
                predicted_price = predict_price(selected_player_type,bowler_model,user_input)
            else:
                predicted_price = "Invalid player type selected"

            st.success(f'Predicted price of {selected_player_type} player is: {predicted_price}')

    if (selected == 'Statistics'):
        st.title('Top 100 Batsmen and Bowlers Visualization')

        st.sidebar.title('Select Rank')
        selected_rank = st.sidebar.slider('POS', min_value=1, max_value=100, value=1)

        st.subheader('Selected Rank Details')

        st.write('**Batsmen Details:**')
        display_player_details(batsmen_data, selected_rank)

        st.write('**Bowlers Details:**')
        display_player_details(bowlers_data, selected_rank)

        if st.button('Visualization for auction'):
            st.title('Auction Visualization')
            plot_top10_buys(auction_top10)
            plot_country_distribution(country_counts)
            plot_role_distribution(role_counts)


        if st.button('Visualization for batsmen and bowlers'):
            st.title('Top 100 Batsmen and Bowlers Visualization')

            display_top_batsmen(batsmen_data)
            display_top_bowlers(bowlers_data)
            display_top_avg_batsmen(batsmen_data)
            plot_batting_avg_distribution(batsmen_data)
            plot_bowling_avg_distribution(bowlers_data)
            plot_runs_vs_avg(batsmen_data)
            plot_wickets_vs_avg(bowlers_data)


if __name__ == "__main__":
    main()
```

10.2 PLAGIARISM REPORT

CHAPTER 11
REFERENCES

Journal References:

[1] Garg, A., & Chaudhary, A. (2023, May). Analysis of IPL Auction Dataset Using Explainable Machine Learning with Lime and H2O AutoML. In 2023 4th International Conference on Intelligent Engineering and Management (ICIEM) (pp. 1-4). IEEE.

[2] Malhotra, G. (2022). A comprehensive approach to predict auction prices and economic value creation of cricketers in the Indian Premier League (IPL). *Journal of Sports Analytics*, *8*(3), 149-170.

[3] Chittibabu, V., & Sundararaman, M. (2023). Base price determination for IPL mega auctions: A player performance-based approach. Journal of Sports Analytics, (Preprint), 1-21.

[4] Das, N. R., Priya, R., Mukherjee, I., & Paul, G. (2021, July). Modified Hedonic based price prediction model for players in IPL auction. In 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE.

[5] Kulkarni, A., Kamath, A. V., Menon, A., Dhatwalia, P., & Rishabh, D. (2020, July). Prediction of player price in IPL auction using machine learning regression algorithms. In 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT) (pp. 1-6). IEEE.

[6] Saikia, H., Bhattacharjee, D., & Bhattacharjee, A. (2013). Performance based market valuation of cricketers in IPL. Sport, Business and Management: An International Journal,3(2), 127-146.

[7] Rastogi, S. K. (2017). What's a Cricketer's Worth? Predicting Bid Prices for Indian Premier League Auctions. Annals of the University Dunarea de Jos of Galati: Fascicle: I, Economics & Applied Informatics, 23(1).

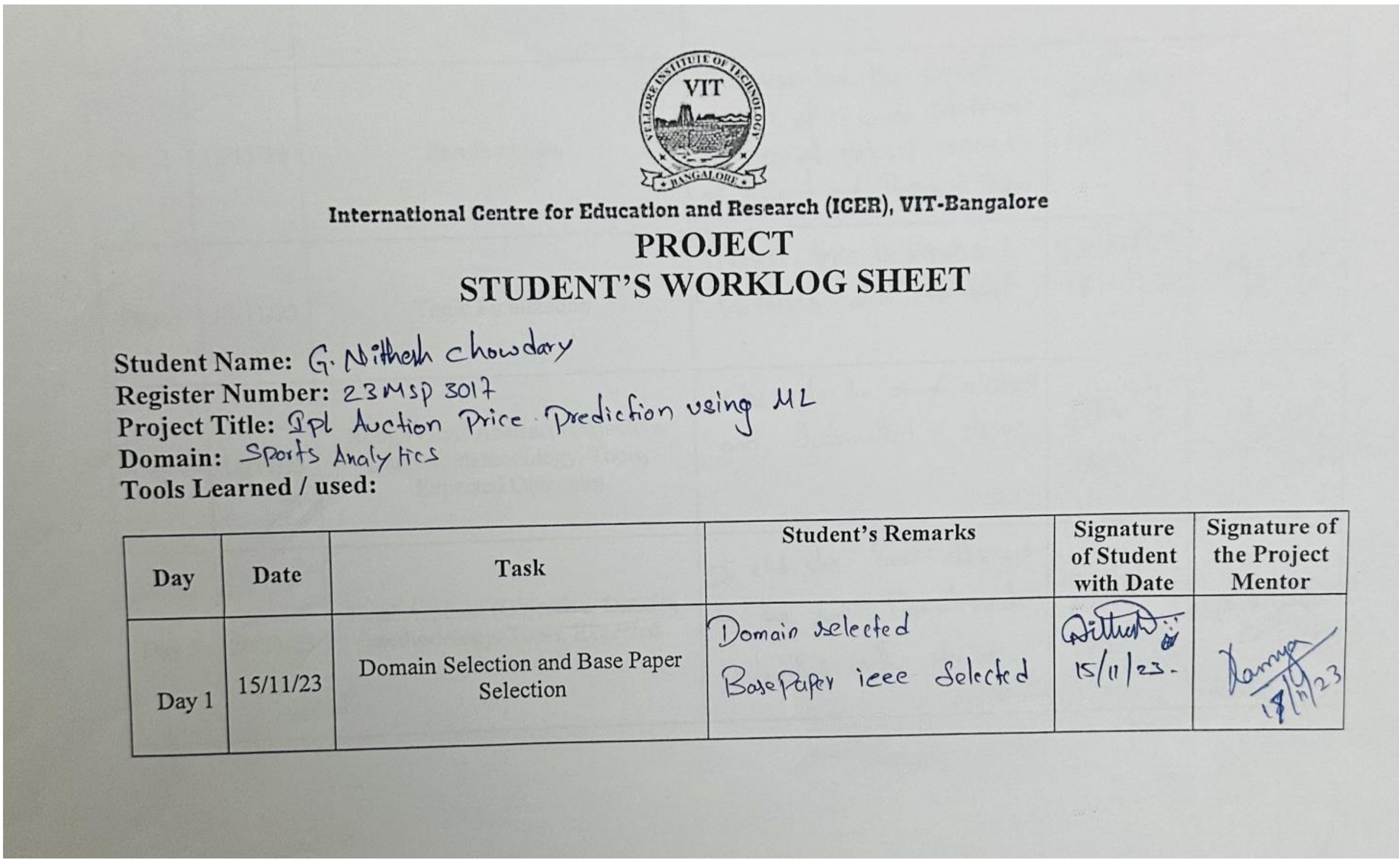
[8]Lamsal, R., & Choudhary, A. (2018). Predicting outcome of Indian premier league (IPL) matches using machine learning. arXiv preprint arXiv:1809.09813.

[9]Saikia, H., Bhattacharjee, D., Mukherjee, D., Saikia, H., Bhattacharjee, D., & Mukherjee, D. (2019). Performance-Based Market Valuation of Cricketers. Cricket Performance Management: Mathematical Formulation and Analytics, 113-128.

[10]Vistro, D. M., Rasheed, F., & David, L. G. (2019). The cricket winner prediction with application of machine learning and data analytics. International journal of scientific & technology Research, 8(09).

CHAPTER 12

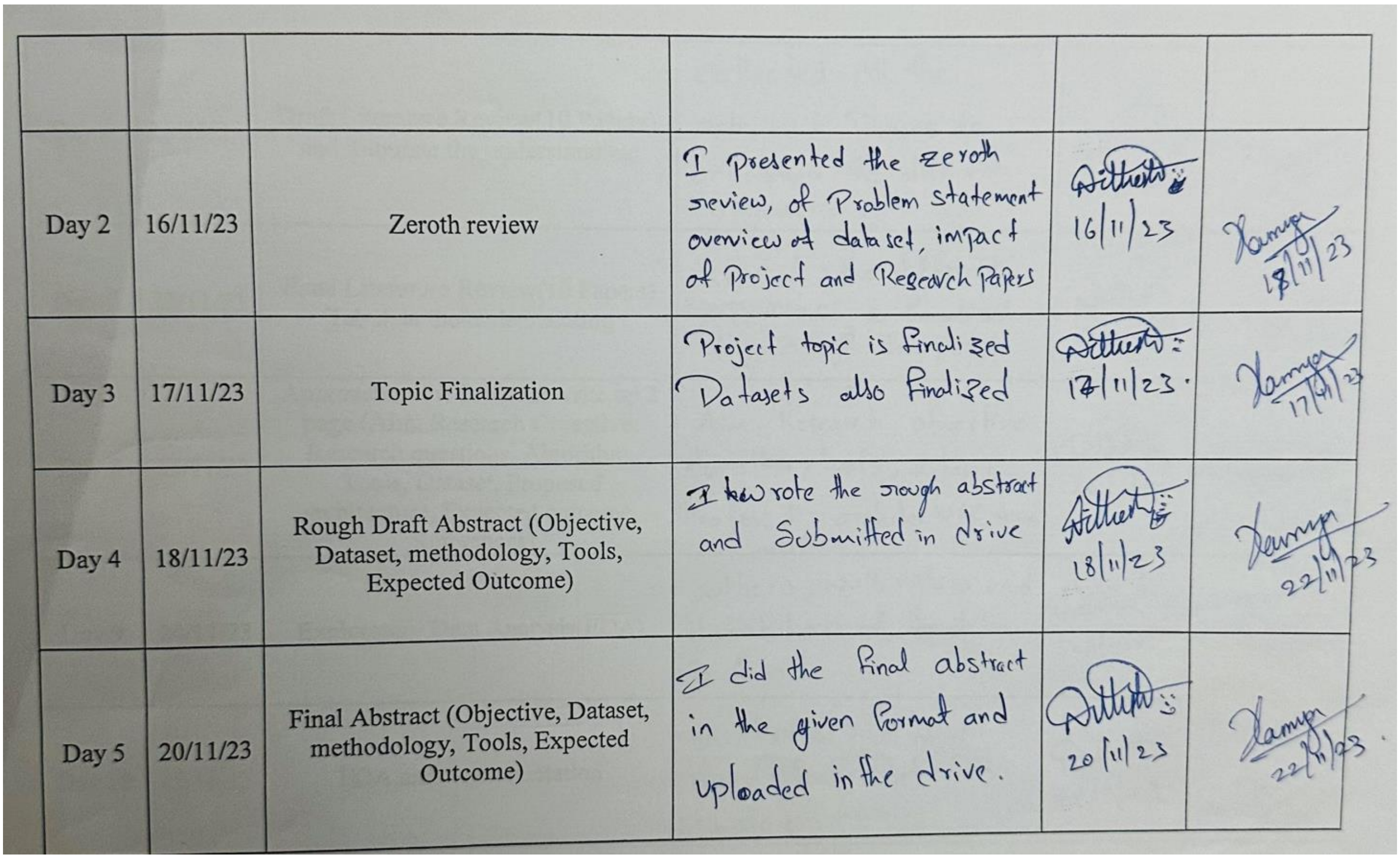
WORKLOG



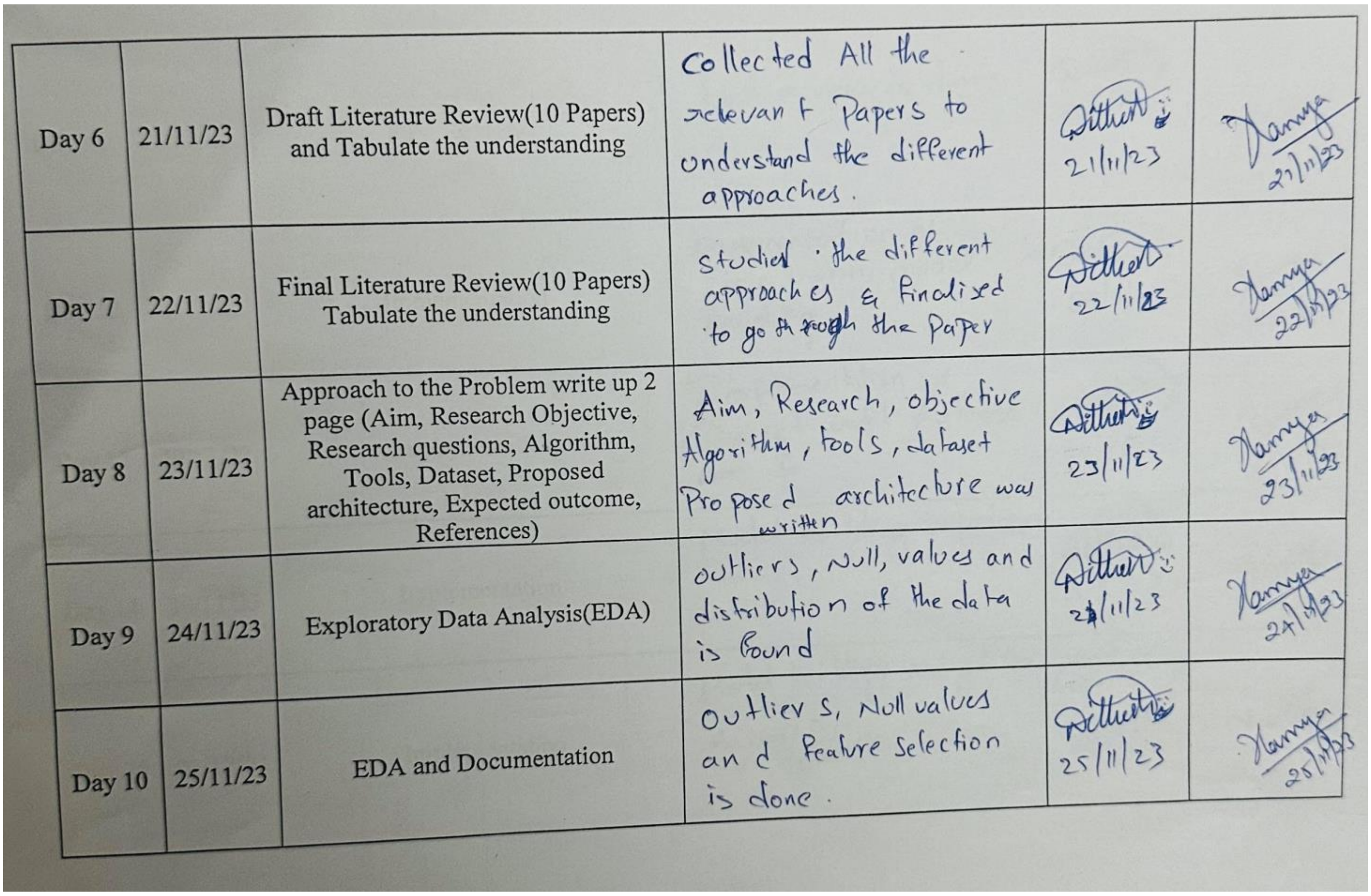
International Centre for Education and Research (ICER), VIT-Bangalore

PROJECT
STUDENT'S WORKLOG SHEET

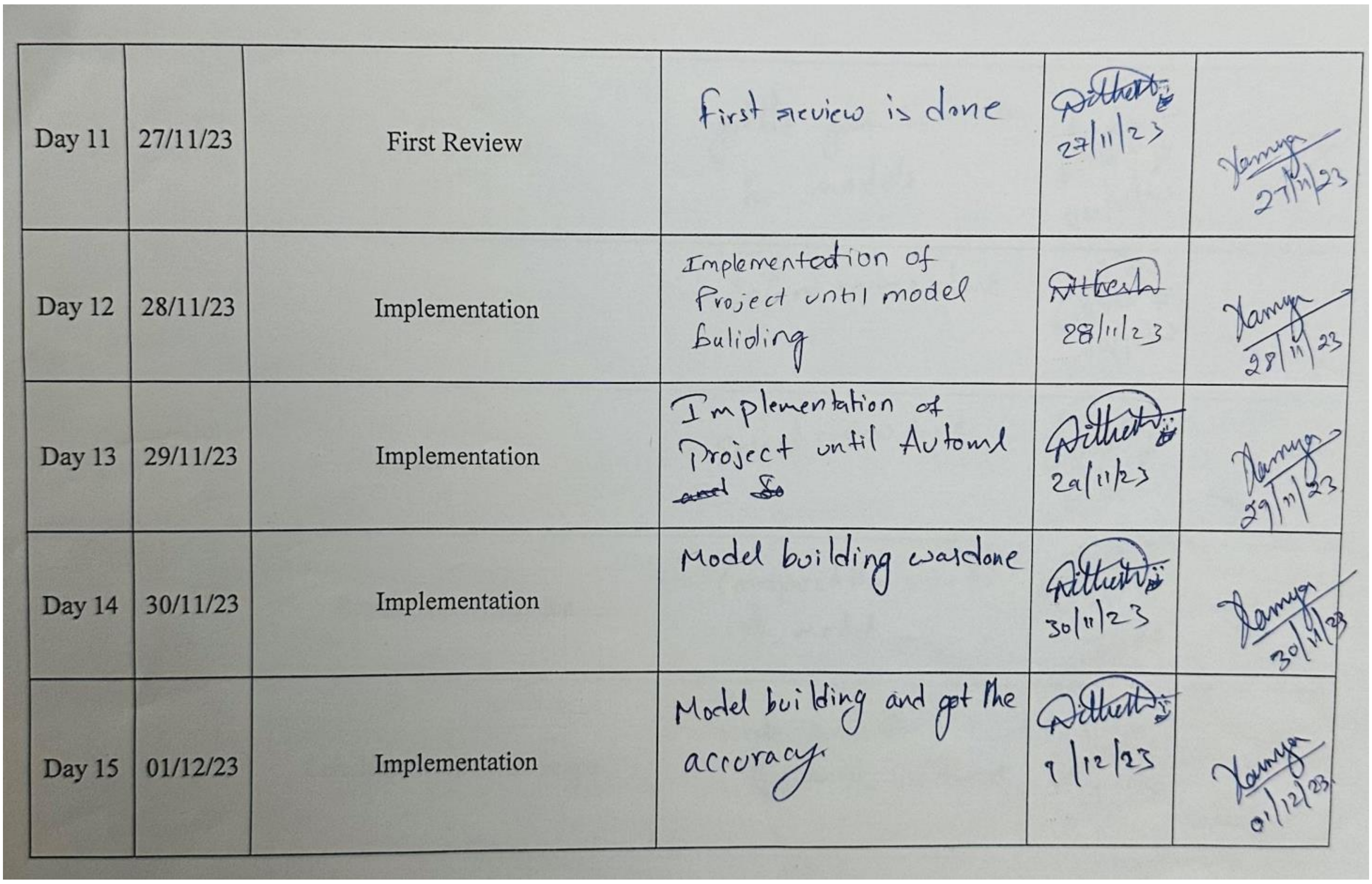
Student Name: G. Nithesh chowdary
Register Number: 23MSP 3017
Project Title: Ipl Auction Price Prediction using ML
Domain: Sports Analytics
Tools Learned / used:



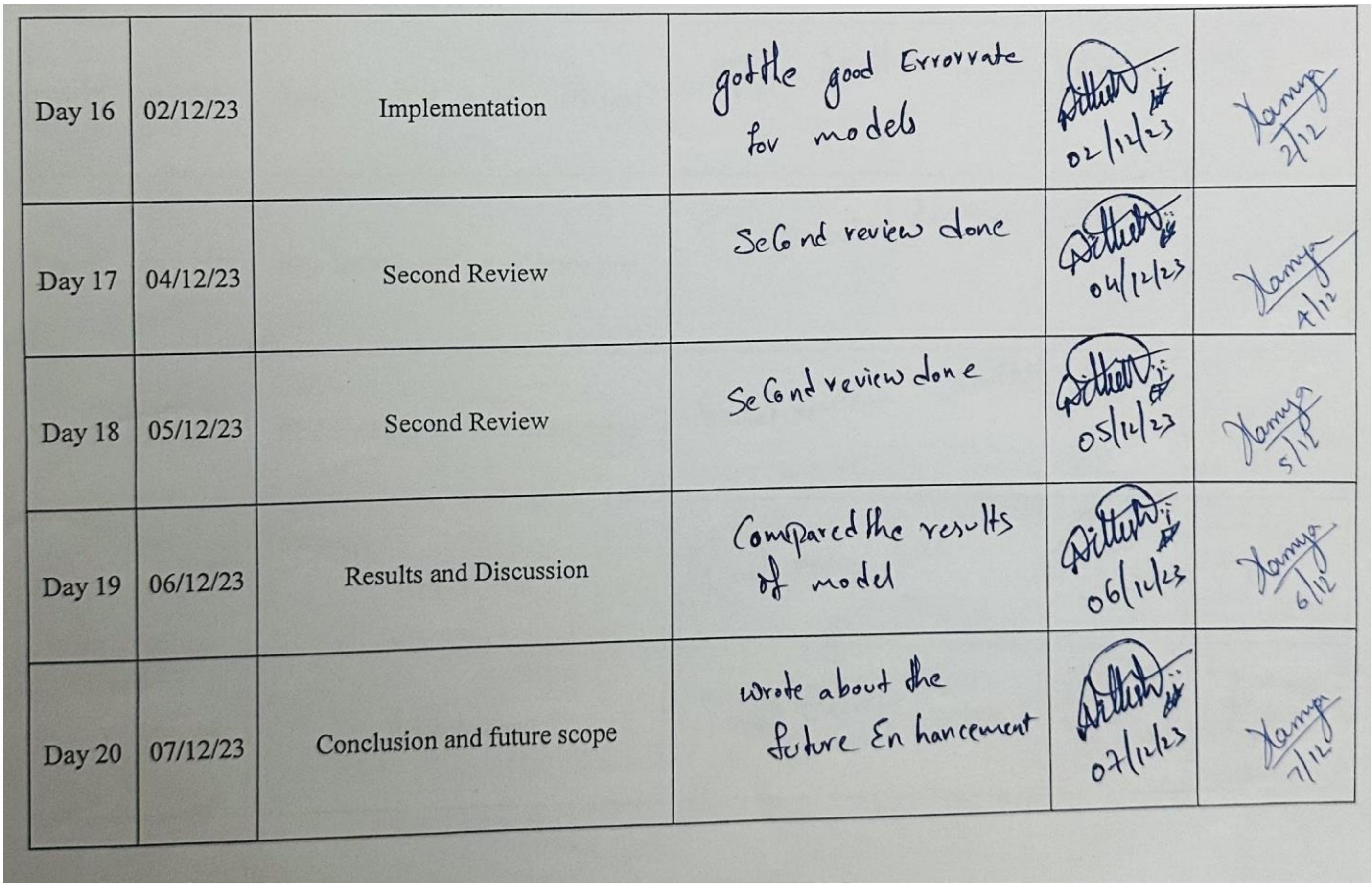
Day	Date	Task	Student's Remarks	Signature of Student with Date	Signature of the Project Mentor
Day 1	15/11/23	Domain Selection and Base Paper Selection	Domain selected BasePaper ieee Selected	Nithesh 15/11/23.	Ramya 18/11/23
Day 2	16/11/23	Zeroth review	I presented the zeroth review, of Problem statement overview of dataset, impact of Project and Research Papers	Nithesh 16/11/23	Ramya 18/11/23
Day 3	17/11/23	Topic Finalization	Project topic is finalized Datasets also finalized	Nithesh 17/11/23.	Ramya 17/11/23
Day 4	18/11/23	Rough Draft Abstract (Objective, Dataset, methodology, Tools, Expected Outcome)	I wrote the rough abstract and Submitted in drive	Nithesh 18/11/23	Ramya 22/11/23
Day 5	20/11/23	Final Abstract (Objective, Dataset, methodology, Tools, Expected Outcome)	I did the final abstract in the given format and uploaded in the drive.	Nithesh 20/11/23	Ramya 22/11/23.



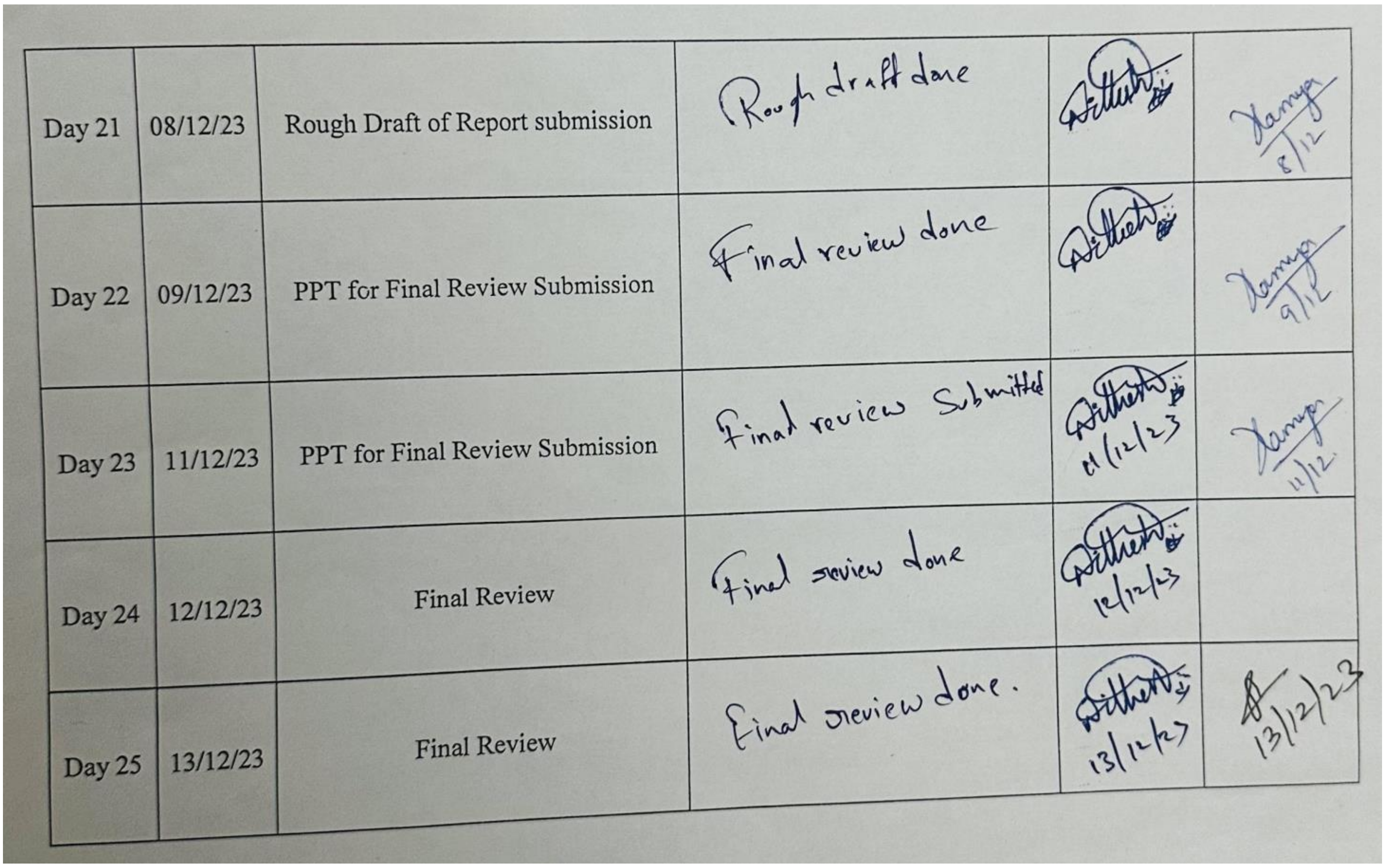
Day 6	21/11/23	Draft Literature Review(10 Papers) and Tabulate the understanding	Collected All the relevant Papers to understand the different approaches.	21/11/23	21/11/23
Day 7	22/11/23	Final Literature Review(10 Papers) Tabulate the understanding	studied the different approaches & finalised to go ~~th~~ through the Paper	22/11/23	22/11/23
Day 8	23/11/23	Approach to the Problem write up 2 page (Aim, Research Objective, Research questions, Algorithm, Tools, Dataset, Proposed architecture, Expected outcome, References)	Aim, Research, objective Algorithm, tools, dataset Proposed architecture was written	23/11/23	23/11/23
Day 9	24/11/23	Exploratory Data Analysis(EDA)	outliers, Null, values and distribution of the data is found	24/11/23	24/11/23
Day 10	25/11/23	EDA and Documentation	Outliers, Null values and feature selection is done.	25/11/23	25/11/23



Day 11	27/11/23	First Review	first review is done	27/11/23	27/11/23
Day 12	28/11/23	Implementation	Implementation of Project until model buliding	28/11/23	28/11/23
Day 13	29/11/23	Implementation	Implementation of Project until Automl ~~and~~	29/11/23	29/11/23
Day 14	30/11/23	Implementation	Model building wasdone	30/11/23	30/11/23
Day 15	01/12/23	Implementation	Model building and got the accuracy.	1/12/23	01/12/23



Day 16	02/12/23	Implementation	gotthe good Errorrate for models	02/12/23	2/12
Day 17	04/12/23	Second Review	SeCond review done	04/12/23	4/12
Day 18	05/12/23	Second Review	SeCond review done	05/12/23	5/12
Day 19	06/12/23	Results and Discussion	Compared the results of model	06/12/23	6/12
Day 20	07/12/23	Conclusion and future scope	wrote about the future Enhancement	07/12/23	7/12



Day 21	08/12/23	Rough Draft of Report submission	Rough draft done		8/12
Day 22	09/12/23	PPT for Final Review Submission	Final review done		9/12
Day 23	11/12/23	PPT for Final Review Submission	Final review Submitted	11/12/23	11/12
Day 24	12/12/23	Final Review	Final sreview done	12/12/23	
Day 25	13/12/23	Final Review	Final sreview done.	13/12/23	13/12/23