

**Cricket Data Analysis and Machine Learning Regression**

**for Score Prediction**

A report submitted in partial fulfilment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

MANEPALLI KALKI NAGA DURGA SAI

REG..NO: 21B91A6229

Blackbucks Pvt.Ltd

(Duration: June 2024 to July 2024)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

S.R.K.R. ENGINEERING COLLEGE

(Autonomous)

SRKR MARG, CHINNA AMIRAM, BHIMAVARAM-534204, A.P

(Recognised by A.I.C.T.E New Delhi) (Accredited by NBA & NAAC)

(Affiliated to JNTU, KAKINADA)



**TABLE OF CONTENTS**

1.ABSTRACT …………………………………………………… 1

2.PROBLEM STATEMENT………………………………………2

3.INTRODUCTION ………………………………………………3

4.PROCEDURE……………………………………………………4

5.CODE…………………………………………………………….8

6. ANALYSIS OF RESULTS…………………………………… 22

7. REFERENCES…………………………………………………22



**ABSTRACT**

This project focuses on applying machine learning (ML) techniques to analyze cricket data and predict match scores. Accurate score prediction is crucial for enhancing strategic decisions in sports analytics, including game strategies, player performance evaluation, and fan engagement. Traditional statistical methods often fall short due to the complex and dynamic nature of cricket matches, making ML an attractive solution.

The primary objective of this project is to develop an advanced score prediction model using state-of-the-art data science methodologies and machine learning algorithms. The study covers essential stages such as data preprocessing, feature engineering, model selection, and evaluation metrics to ensure the reliability and accuracy of the predictions. We utilized authentic match data, including player statistics, team dynamics, pitch conditions, and weather, to simulate various match scenarios.

The results demonstrate the potential of ML-based systems in predicting cricket scores with high accuracy and minimal errors. Furthermore, continuous model monitoring and optimization are essential to maintaining the system's effectiveness over time, providing valuable insights for teams, analysts, and fans alike.

1



**PROBLEM STATEMENT**

Cricket, a globally celebrated sport, heavily influences the lives of millions, shaping cultural, social, and economic aspects in various countries. Accurate analysis and prediction of cricket match outcomes are pivotal for teams, players, fans, and analysts alike. Traditional methods of predicting match scores often rely on simplistic approaches or historical averages, which may not capture the complex dynamics of modern cricket matches. Addressing this challenge necessitates the development of a sophisticated model capable of providing precise score predictions and insightful analyses of cricket data.

The focus of this project is to create an advanced Cricket Data Analysis and Score Prediction model using a comprehensive dataset that includes a wide range of cricketing parameters. This dataset, derived from multiple sources such as match records, player statistics, and game conditions, encompasses a diverse range of matches played across different venues, formats, and conditions. The model must navigate through intricate patterns within this data, considering numerous factors such as runs scored, wickets taken, overs bowled, player performance, and match context.

The proposed model must analyze these multifaceted parameters to identify trends and make accurate predictions about match outcomes. By leveraging advanced machine learning algorithms and data science techniques, the model aims to uncover the subtleties embedded within the cricketing data. This includes detecting patterns in player performances, understanding the impact of various match conditions, and evaluating the interactions between different factors that contribute to the overall match results.

In conclusion, as we delve into the complex domain of cricket data analysis and score prediction, developing a robust and adaptive model becomes crucial. Integrating advanced analytics, machine learning, and a comprehensive consideration of diverse cricketing factors is essential to enhance the accuracy of match predictions and provide valuable insights. This endeavor is vital to support teams' strategic decisions, engage fans with more accurate forecasts, and optimize performance analysis across the sport.

2



# INTRODUCTION

Data science is a multidisciplinary field that combines various techniques, algorithms, processes, and systems to extract knowledge and insights from structured and unstructured data. It has emerged as a crucial discipline in today's digital age, where data is generated at an unprecedented rate across diverse industries and domains.

Key Components of Data Science:

1. Data collection

2. Data cleaning and preprocessing

3. Exploratory Data Analysis

4. Feature Engineering

5. Machine Leaning

6. Model Evaluation

7. Data visualization

8. Big data and Cloud computing

9. Artificial Intelligence

10. Ethics and Privacy

**MACHINE LEARNING**

Machine learning is a subset of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to learn and make predictions or decisions without being explicitly programmed.

Three types of Machine learning:

1. Supervised Learning : In supervised learning, models are trained on labeled data, where the input data is paired with the correct output. The algorithm learns to map inputs to outputs, making it suitable for tasks like classification and regression.

2. Unsupervised Learning : Unsupervised learning involves working with unlabelled data to discover patterns

and structure within the data. Common tasks include clustering and dimensionality reduction.

3. Reinforcement learning : Reinforcement learning is concerned with training agents to make sequential

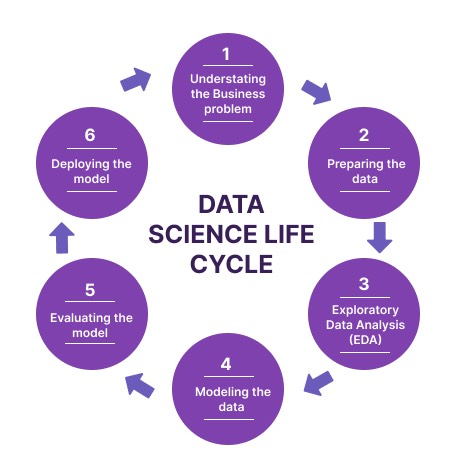
decisions in an environment to maximise a reward.It is often used in applications like game playing and

robotics.

**DATA SCIENCE**

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data.

3



**PROCEDURE**

Data Science Life

4



**1. Understanding the Business Problem**

This is the foundational step in the data science life cycle. It involves:

* **Identifying Objectives:** Understanding the goals and objectives of the project from a business perspective. This may involve discussions with stakeholders to identify key questions that need answers.
* **Defining Scope:** Clearly defining the scope of the project, including what is and isn’t included.
* **Determining Requirements:** Gathering detailed requirements and constraints, such as data availability, resource limitations, and deadlines.

**2. Preparing the Data**

This step involves all activities required to prepare the data for analysis. Key tasks include:

* **Data Collection:** Gathering data from various sources which could include databases, APIs, web scraping, or other methods.
* **Data Cleaning:** Identifying and rectifying errors and inconsistencies in the data, such as missing values, outliers, or incorrect formats.
* **Data Integration:** Combining data from different sources into a unified dataset.
* **Data Transformation:** Modifying data to fit the required format for analysis, which can involve normalising, aggregating, or encoding variables.

**3. Exploratory Data Analysis (EDA)**

EDA is a crucial step to understand the underlying patterns and characteristics of the data. It involves:

* **Descriptive Statistics:** Calculating metrics like mean, median, mode, standard deviation, etc., to summarise the data.
* **Data Visualization:** Creating plots and charts (e.g., histograms, scatter plots, box plots) to visually inspect data distributions and relationships.
* **Identifying Patterns:** Looking for trends, correlations, or anomalies that may influence the analysis.

**4. Modelling the Data**

In this step, predictive or descriptive models are created based on the prepared data. This involves:

* **Selecting Algorithms:** Choosing appropriate machine learning or statistical algorithms based on the problem and data characteristics.
* **Training Models:** Applying algorithms to the training data to develop models.
* **Validation:** Evaluating models using a separate validation set to assess their performance.

5



## Algorithms Used :

* Random Forest
* Decision Tree
* Linear Regression
* XGBoost

## Random Forest

The Random Forest algorithm is highly effective for weather prediction due to its robustness and accuracy. It combines multiple decision trees to mitigate overfitting, providing a more generalised model. Each tree is trained on different data subsets, and predictions are based on the majority vote, leveraging the wisdom of the crowd to improve accuracy. This is particularly useful for complex weather datasets with numerous features. Additionally, Random Forest handles both numerical and categorical data and offers feature importance scores, helping identify the most influential factors affecting weather patterns.

## Decision Tree

Decision Trees are powerful models for weather prediction due to their ease of interpretation and ability to model nonlinear relationships. Each node represents a decision point, effectively capturing interactions between meteorological factors like temperature, humidity, wind speed, and pressure. They require minimal preprocessing, handle missing values naturally, and are quick to train. These qualities make Decision Trees a practical choice for rapid deployment and real-time weather prediction scenarios.

## Linear Regression

Linear regression models the relationship between a dependent variable and one or more independent variables by fitting a straight line through the data. It assumes a linear relationship, making it simple and easy to interpret. This technique is widely used for prediction and analysis in various fields. However, it is limited by its linearity assumption and may not capture more complex patterns. Despite this, linear regression is a foundational tool in statistics and machine learning

## XGBoost Regressor

XGBoost, short for "Extreme Gradient Boosting," is a powerful machine learning algorithm based on the gradient boosting framework. It is designed for speed and performance, making it popular for structured or tabular data. XGBoost builds an ensemble of decision trees sequentially, where each tree corrects the errors of the previous ones, optimizing for a specified loss function. It employs techniques like regularization to prevent overfitting and includes efficient handling of missing values and sparse data. Known for its scalability, XGBoost performs well on both small and large datasets, often winning in data science competitions. Its combination of accuracy, flexibility y, and efficiency makes it a go-to choice for many predictive modeling tasks.

6



**5. Evaluating the Model**

Once models are built, their performance must be rigorously evaluated to ensure they meet the business requirements. This includes:

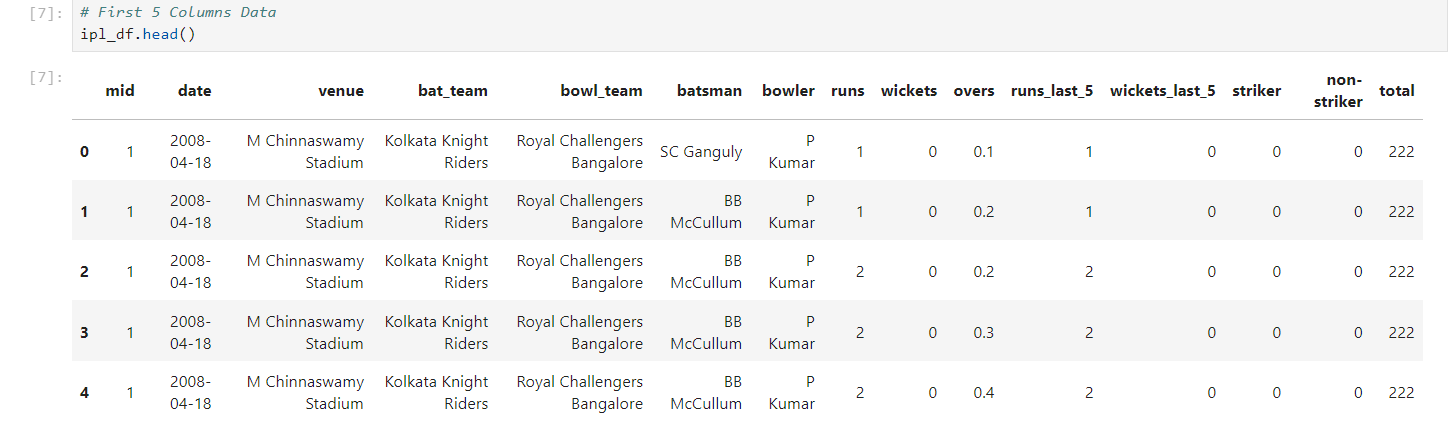
* **Performance Metrics:** Calculating metrics like accuracy, precision, recall, F1 score, ROC-AUC for classification models, or RMSE, MAE for regression models.
* **Cross-Validation:** Using techniques like k-fold cross-validation to ensure model reliability.
* **Comparison:** Comparing different models to select the best one.
* **Interpretability:** Ensuring the model’s predictions can be understood and justified.

**6. Deploying the Model**

The final step is to deploy the model into a production environment where it can be used to make real-time decisions. Key activities include:

* **Model Integration:** Integrating the model with existing systems and workflows.
* **Monitoring:** Continuously monitoring model performance to ensure it remains accurate over time.
* **Updating:** Regularly updating the model with new data to maintain its performance.
* **Documentation:** Documenting the entire process and model characteristics for future reference and compliance.

7



# PROGRAMMING CODE

**Importing the required libraries**

Imports necessary libraries such as numpy, pandas, seaborn, etc. for data manipulation,

visualization, statistical analysis, and machine learning.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.tree import DecisionTreeRegressor

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from xgboost import XGBRegressor

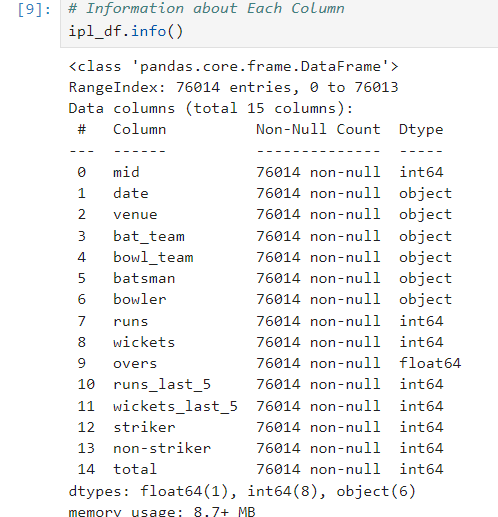
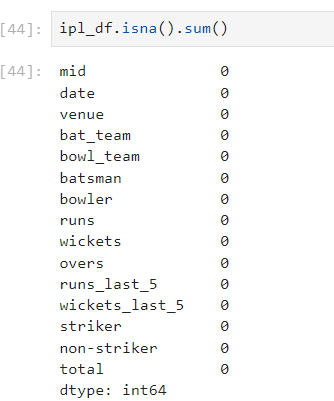
**Loading the dataset**

ipl\_df = pd.read\_csv('ipl\_data.csv')

ipl\_df.head()

8

8



# Data Preprocessing

print(f"Dataset successfully Imported of Shape : {ipl\_df.shape}")

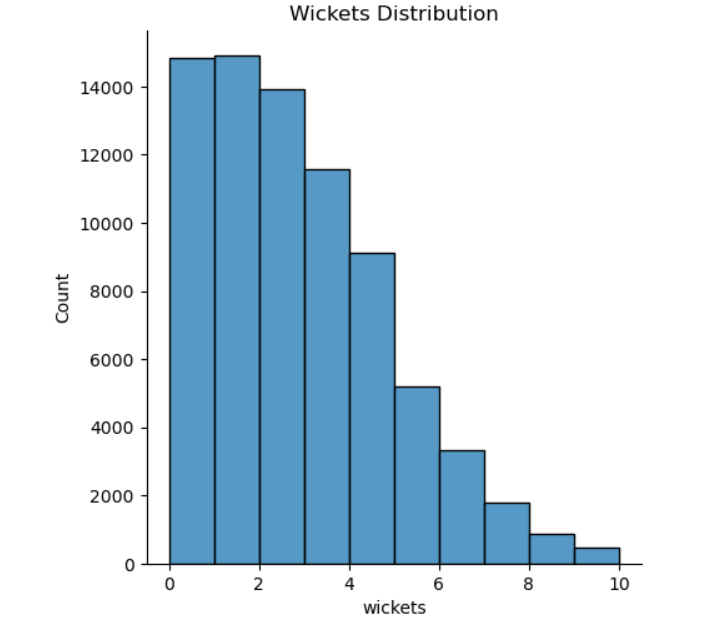
ipl\_df.isna().sum()

There are no Null values in the dataset

ipl\_df.info()

9

d1.info()



import seaborn as sns

import matplotlib.pyplot as plt

sns.displot(ipl\_df['wickets'],kde=False,bins=10)

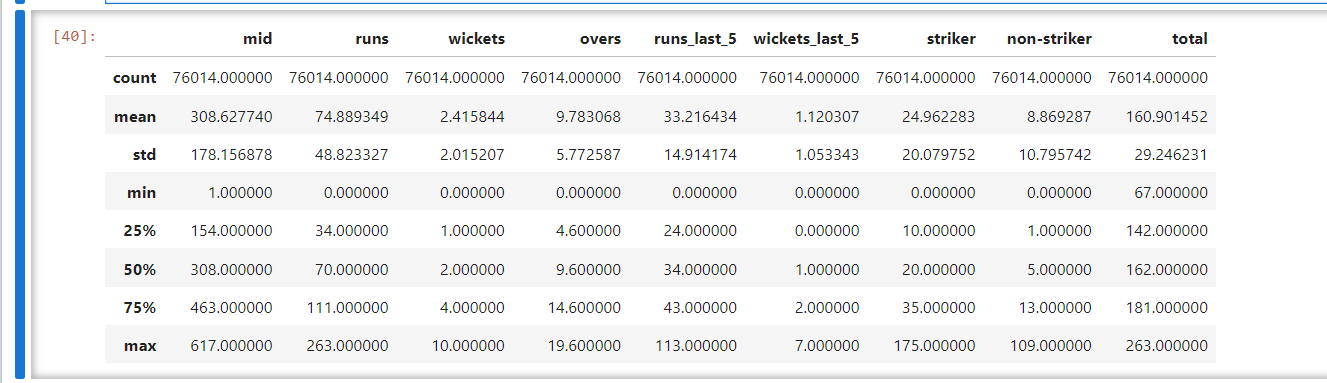
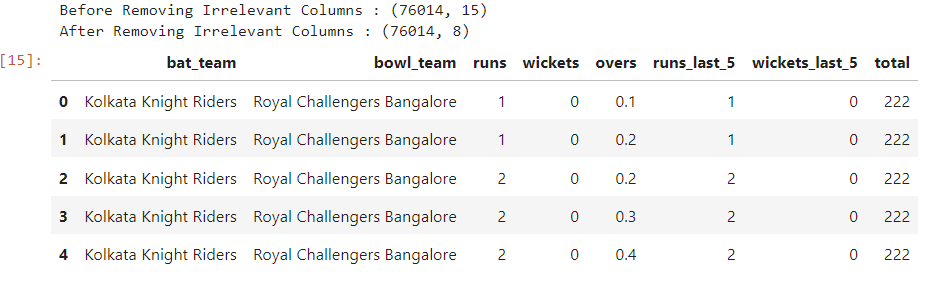
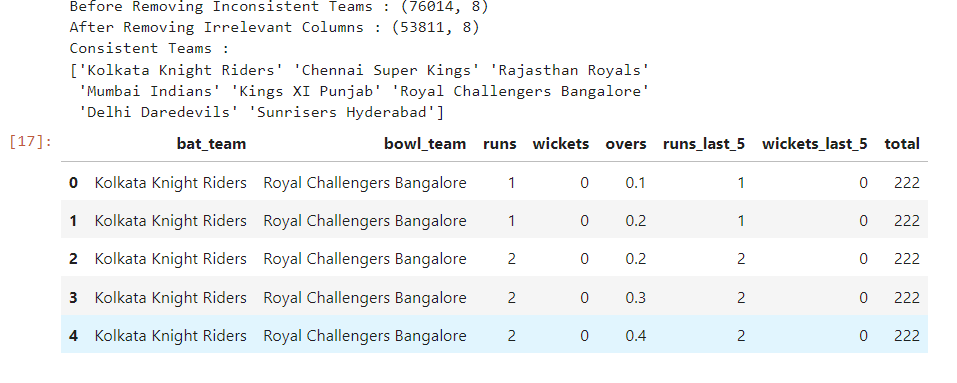
plt.title("Wickets Distribution")

plt.show()

10

10

sns.countplot(x='weather', data=data , palette='hls')



ipl\_df.describe()

ipl\_df.columns

irrelevant = ['mid', 'date', 'venue','batsman', 'bowler', 'striker', 'non-striker']

print(f'Before Removing Irrelevant Columns : {ipl\_df.shape}')

ipl\_df = ipl\_df.drop(irrelevant, axis=1) # Drop Irrelevant Columns

print(f'After Removing Irrelevant Columns : {ipl\_df.shape}')

ipl\_df.head()

const\_teams = ['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',

'Mumbai Indians', 'Kings XI Punjab', 'Royal Challengers Bangalore',

'Delhi Daredevils', 'Sunrisers Hyderabad']

print(f'Before Removing Inconsistent Teams : {ipl\_df.shape}')

ipl\_df = ipl\_df[(ipl\_df['bat\_team'].isin(const\_teams)) & (ipl\_df['bowl\_team'].isin(const\_teams))]

print(f'After Removing Irrelevant Columns : {ipl\_df.shape}')

print(f"Consistent Teams : \n{ipl\_df['bat\_team'].unique()}")

ipl\_df.head()

11



numeric\_columns = ipl\_df.select\_dtypes(include=[float, int])

corr\_matrix = numeric\_columns.corr()

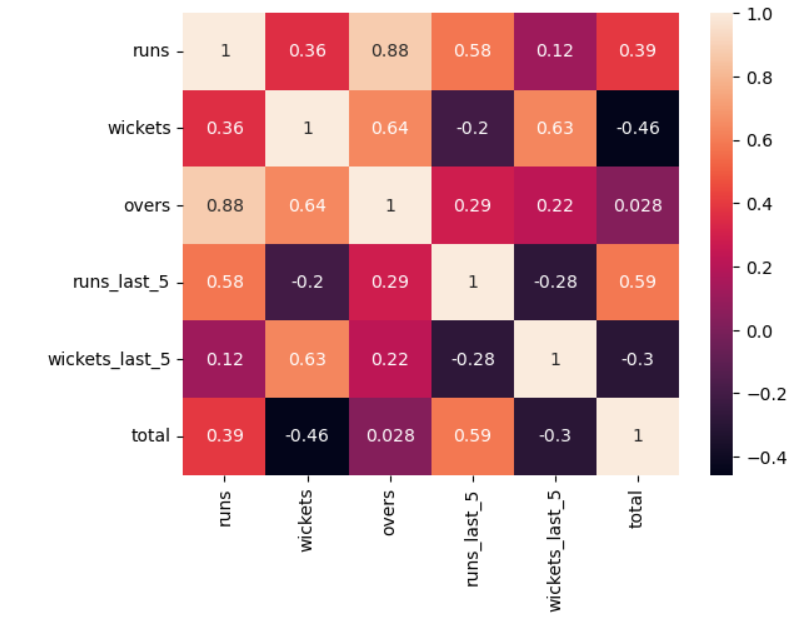
# Plotting the heatmap

import seaborn as sns

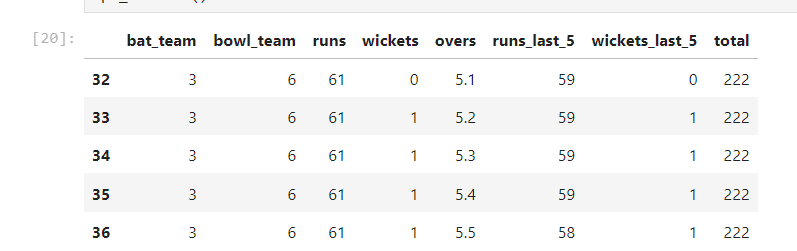
import matplotlib.pyplot as plt

sns.heatmap(corr\_matrix, annot=True)

plt.show()



12



from sklearn.preprocessing import LabelEncoder, OneHotEncoder

le = LabelEncoder()

for col in ['bat\_team', 'bowl\_team']:

ipl\_df[col] = le.fit\_transform(ipl\_df[col])

ipl\_df.head()

cols = ['batting\_team\_Chennai Super Kings', 'batting\_team\_Delhi Daredevils', 'batting\_team\_Kings XI Punjab',

'batting\_team\_Kolkata Knight Riders', 'batting\_team\_Mumbai Indians', 'batting\_team\_Rajasthan Royals',

'batting\_team\_Royal Challengers Bangalore', 'batting\_team\_Sunrisers Hyderabad',

'bowling\_team\_Chennai Super Kings', 'bowling\_team\_Delhi Daredevils', 'bowling\_team\_Kings XI Punjab',

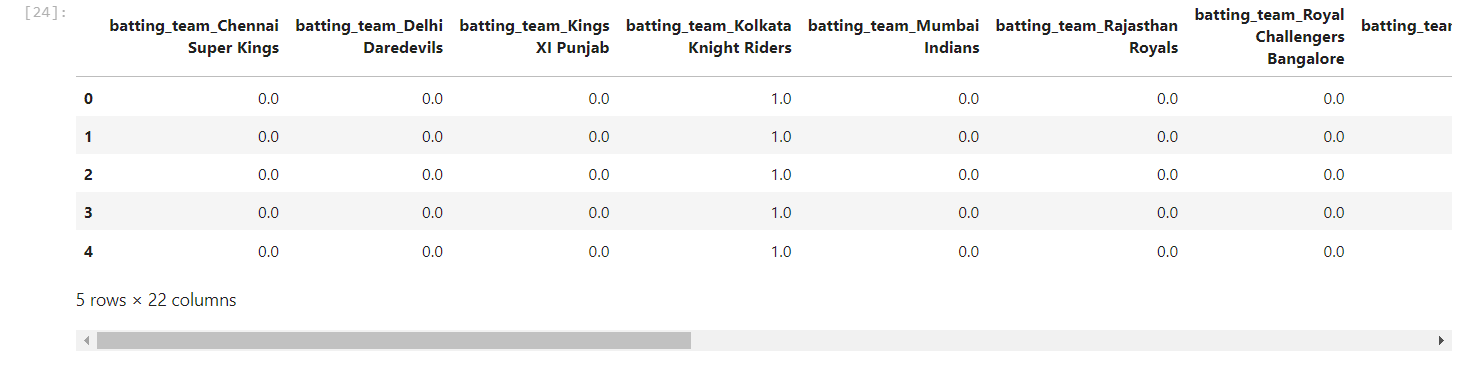
'bowling\_team\_Kolkata Knight Riders', 'bowling\_team\_Mumbai Indians', 'bowling\_team\_Rajasthan Royals',

'bowling\_team\_Royal Challengers Bangalore', 'bowling\_team\_Sunrisers Hyderabad', 'runs', 'wickets', 'overs',

'runs\_last\_5', 'wickets\_last\_5', 'total']

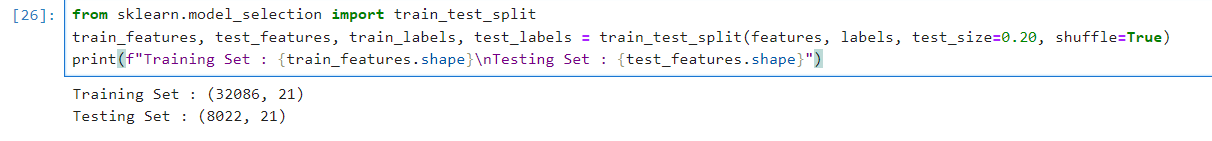
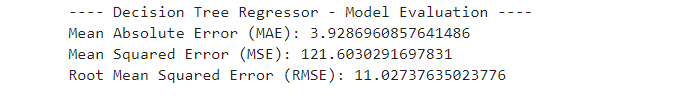
df = pd.DataFrame(ipl\_df, columns=cols)

df.head()



13

13



Prepare Train and Test Data

features = df.drop(['total'], axis=1)

labels = df['total']

from sklearn.model\_selection import train\_test\_split

train\_features, test\_features, train\_labels, test\_labels = train\_test\_split(features, labels, test\_size=0.20, shuffle=True)

print(f"Training Set : {train\_features.shape}\nTesting Set : {test\_features.shape}")

**ML Algorithms**

**1.** **Decision Tree Regressor**

from sklearn.tree import DecisionTreeRegressor

tree = DecisionTreeRegressor()

# Train Model

tree.fit(train\_features, train\_labels)

# Evaluate Model

train\_score\_tree = str(tree.score(train\_features, train\_labels) \* 100)

test\_score\_tree = str(tree.score(test\_features, test\_labels) \* 100)

print(f'Train Score : {train\_score\_tree[:5]}%\nTest Score : {test\_score\_tree[:5]}%')

models["tree"] = test\_score\_tree

Train Score : 99.99%

Test Score : 86.11%

from sklearn.metrics import mean\_absolute\_error as mae, mean\_squared\_error as mse

print("---- Decision Tree Regressor - Model Evaluation ----")

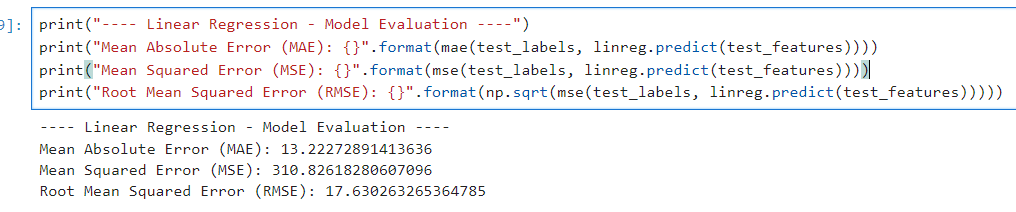
print("Mean Absolute Error (MAE): {}".format(mae(test\_labels, tree.predict(test\_features))))

print("Mean Squared Error (MSE): {}".format(mse(test\_labels, tree.predict(test\_features))))

print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(test\_labels, tree.predict(test\_features)))))

14

14



**2.** **Linear Regression**

from sklearn.linear\_model import LinearRegression

linreg = LinearRegression()

# Train Model

linreg.fit(train\_features, train\_labels)

train\_score\_linreg = str(linreg.score(train\_features, train\_labels) \* 100)

test\_score\_linreg = str(linreg.score(test\_features, test\_labels) \* 100)

print(f'Train Score : {train\_score\_linreg[:5]}%\nTest Score : {test\_score\_linreg[:5]}%')

models["linreg"] = test\_score\_linreg

Train Score : 65.85%

Test Score : 66.15%

print("---- Linear Regression - Model Evaluation ----")

print("Mean Absolute Error (MAE): {}".format(mae(test\_labels, linreg.predict(test\_features))))

print("Mean Squared Error (MSE): {}".format(mse(test\_labels, linreg.predict(test\_features))))

print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(test\_labels, linreg.predict(test\_features)))))

15

15



**3.** **Random Forest Regression**

from sklearn.ensemble import RandomForestRegressor

forest = RandomForestRegressor()

# Train Model

forest.fit(train\_features, train\_labels)

# Evaluate Model

train\_score\_forest = str(forest.score(train\_features, train\_labels)\*100)

test\_score\_forest = str(forest.score(test\_features, test\_labels)\*100)

print(f'Train Score : {train\_score\_forest[:5]}%\nTest Score : {test\_score\_forest[:5]}%')

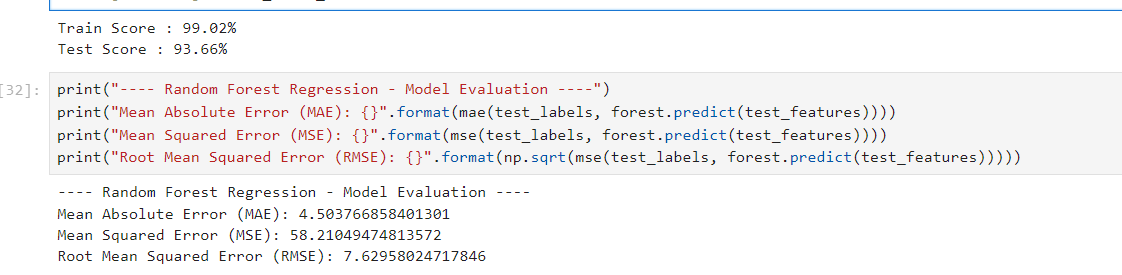
models["forest"] = test\_score\_forest

print("---- Random Forest Regression - Model Evaluation ----")

print("Mean Absolute Error (MAE): {}".format(mae(test\_labels, forest.predict(test\_features))))

print("Mean Squared Error (MSE): {}".format(mse(test\_labels, forest.predict(test\_features))))

print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(test\_labels, forest.predict(test\_features)))))



**4.** **Support Vector Machine**

from sklearn.svm import SVR

svm = SVR()

# Train Model

svm.fit(train\_features, train\_labels)

train\_score\_svm = str(svm.score(train\_features, train\_labels)\*100)

test\_score\_svm = str(svm.score(test\_features, test\_labels)\*100)

print(f'Train Score : {train\_score\_svm[:5]}%\nTest Score : {test\_score\_svm[:5]}%')

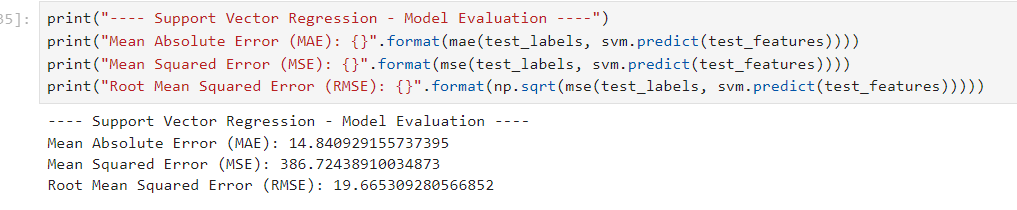
models["svm"] = test\_score\_svm

Train Score : 57.31%

Test Score : 57.89%

16

16



**5.** **XGBoost**

from xgboost import XGBRegressor

xgb = XGBRegressor()

# Train Model

xgb.fit(train\_features, train\_labels)

train\_score\_xgb = str(xgb.score(train\_features, train\_labels)\*100)

test\_score\_xgb = str(xgb.score(test\_features, test\_labels)\*100)

print(f'Train Score : {train\_score\_xgb[:5]}%\nTest Score : {test\_score\_xgb[:5]}%')

models["xgb"] = test\_score\_xgb

Train Score : 88.40%

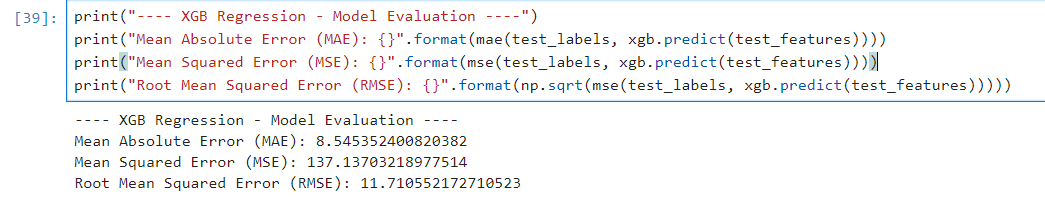
Test Score : 85.06%

print("---- XGB Regression - Model Evaluation ----")

print("Mean Absolute Error (MAE): {}".format(mae(test\_labels, xgb.predict(test\_features))))

print("Mean Squared Error (MSE): {}".format(mse(test\_labels, xgb.predict(test\_features))))

print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(test\_labels, xgb.predict(test\_features)))))



17

17



**6.** **KNR**

from sklearn.neighbors import KNeighborsRegressor

knr = KNeighborsRegressor()

# Train Model

knr.fit(train\_features, train\_labels)

train\_score\_knr = str(knr.score(train\_features, train\_labels)\*100)

test\_score\_knr = str(knr.score(test\_features, test\_labels)\*100)

print(f'Train Score : {train\_score\_knr[:5]}%\nTest Score : {test\_score\_knr[:5]}%')

models["knr"] = test\_score\_knr

Train Score : 86.66%

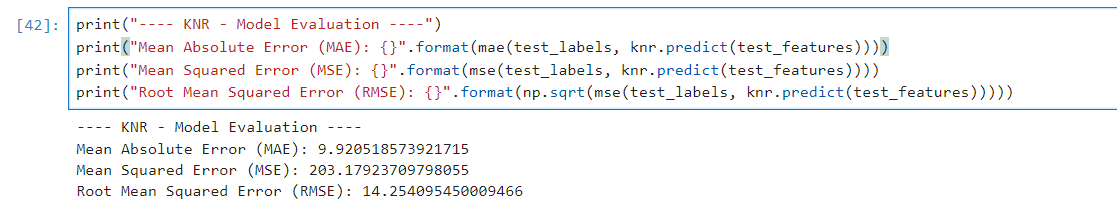
Test Score : 77.87%

print("---- KNR - Model Evaluation ----")

print("Mean Absolute Error (MAE): {}".format(mae(test\_labels, knr.predict(test\_features))))

print("Mean Squared Error (MSE): {}".format(mse(test\_labels, knr.predict(test\_features))))

print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(test\_labels, knr.predict(test\_features)))))



**Best Model**

import matplotlib.pyplot as plt

model\_names = list(models.keys())

accuracy = list(map(float, models.values()))

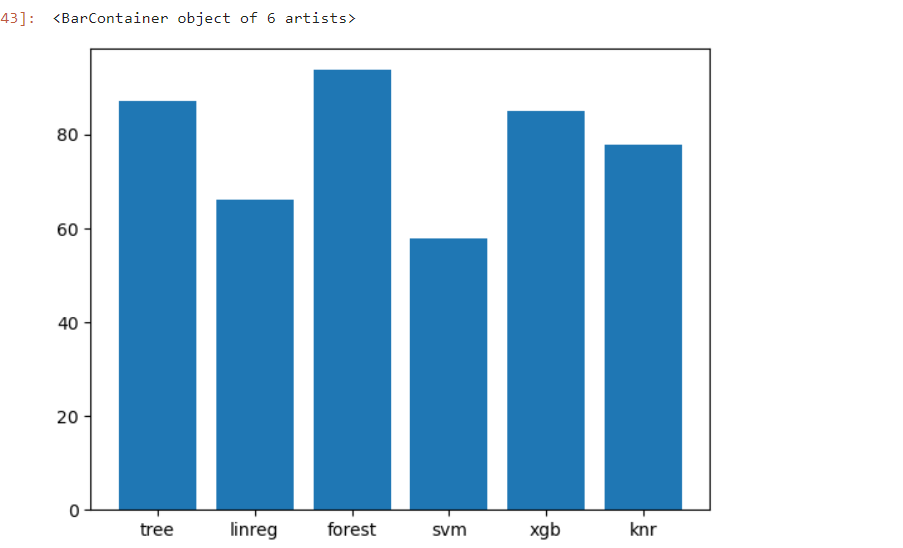
# creating the bar plot

plt.bar(model\_names, accuracy)

18

18





**Predictions**

def score\_predict(batting\_team, bowling\_team, runs, wickets, overs, runs\_last\_5, wickets\_last\_5, model=forest):

prediction\_array = []

# Batting Team

if batting\_team == 'Chennai Super Kings':

prediction\_array = prediction\_array + [1,0,0,0,0,0,0,0]

elif batting\_team == 'Delhi Daredevils':

prediction\_array = prediction\_array + [0,1,0,0,0,0,0,0]

elif batting\_team == 'Kings XI Punjab':

prediction\_array = prediction\_array + [0,0,1,0,0,0,0,0]

elif batting\_team == 'Kolkata Knight Riders':

prediction\_array = prediction\_array + [0,0,0,1,0,0,0,0]

elif batting\_team == 'Mumbai Indians':

prediction\_array = prediction\_array + [0,0,0,0,1,0,0,0]

elif batting\_team == 'Rajasthan Royals':

prediction\_array = prediction\_array + [0,0,0,0,0,1,0,0]

elif batting\_team == 'Royal Challengers Bangalore':

prediction\_array = prediction\_array + [0,0,0,0,0,0,1,0]

elif batting\_team == 'Sunrisers Hyderabad':

prediction\_array = prediction\_array + [0,0,0,0,0,0,0,1]

# Bowling Team

if bowling\_team == 'Chennai Super Kings':

prediction\_array = prediction\_array + [1,0,0,0,0,0,0,0]

elif bowling\_team == 'Delhi Daredevils':

prediction\_array = prediction\_array + [0,1,0,0,0,0,0,0]

elif bowling\_team == 'Kings XI Punjab':

prediction\_array = prediction\_array + [0,0,1,0,0,0,0,0]

elif bowling\_team == 'Kolkata Knight Riders':

prediction\_array = prediction\_array + [0,0,0,1,0,0,0,0]

elif bowling\_team == 'Mumbai Indians':

prediction\_array = prediction\_array + [0,0,0,0,1,0,0,0]

elif bowling\_team == 'Rajasthan Royals':

prediction\_array = prediction\_array + [0,0,0,0,0,1,0,0]

19

elif bowling\_team == 'Royal Challengers Bangalore':

prediction\_array = prediction\_array + [0,0,0,0,0,0,1,0]

elif bowling\_team == 'Sunrisers Hyderabad':

prediction\_array = prediction\_array + [0,0,0,0,0,0,0,1]

prediction\_array = prediction\_array + [runs, wickets, overs, runs\_last\_5, wickets\_last\_5]

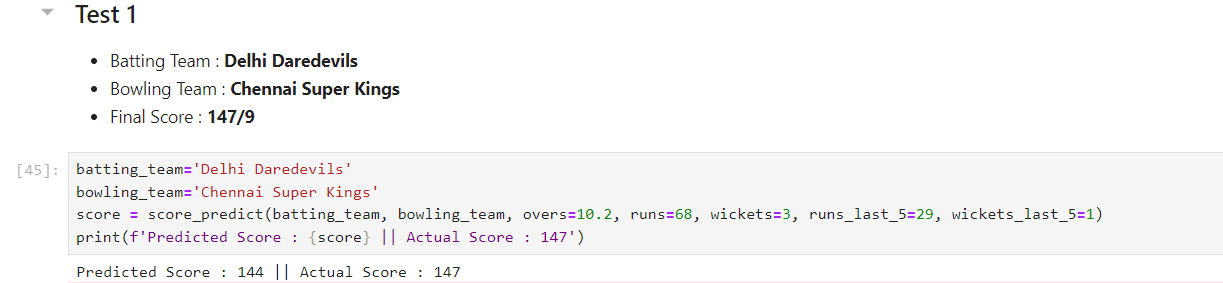
prediction\_array = np.array([prediction\_array])

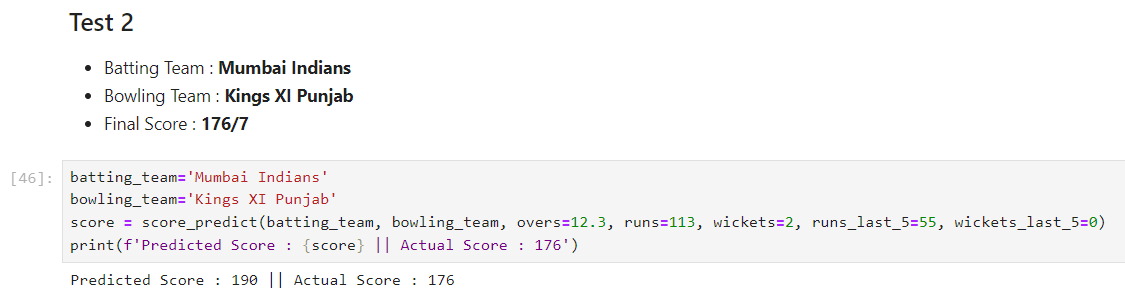
pred = model.predict(prediction\_array)

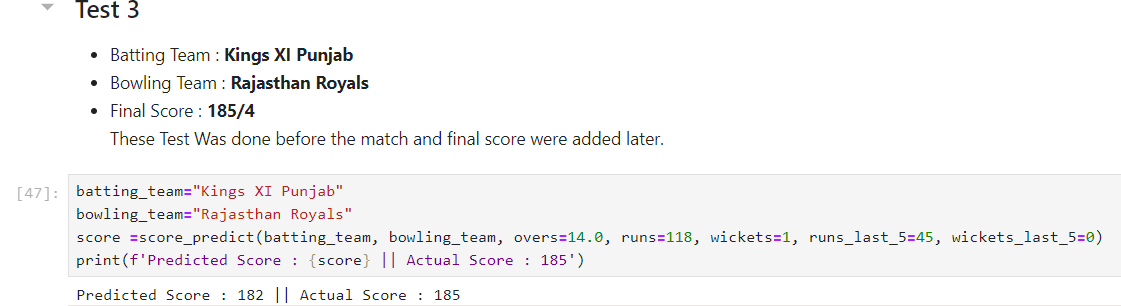
return int(round(pred[0]))

19









20

20



**ANALYSIS OF RESULTS**

**Random Forest**

* Accuracy 93.66%

The Random Fores model achieved an accuracy of 93.6%. This indicates that the model is well-suited for the problem at hand, with strong generalisation capabilities on the testing set.

**Decision Tree**

* Accuracy: 86.11%

The Decision Tree model has an accuracy of 86.11%. While this is lower than the Random Forest, it still shows a reasonable performance. This suggests that the model may be slightly overfitting to the training data.

**Linear Regression**

* Accuracy: 66.15%

The Linear Regression achieved the highest accuracy of 66.15%. %. While this is lower than the Random Forest, it still shows a reasonable performance.

**KNN Classifier**

* Accuracy: 77.7%

The KNN Classifier has an accuracy of 72.7%. This performance is similar to the Decision Tree, suggesting that while the model is capable of capturing patterns in the data, it may not generalise as well as the Naive-Bayes Classifier or Random Forest.

**Summary**

Among the evaluated models, the Random Forest demonstrated the highest accuracy, followed by the Decision Tree model. The Linear Regression and KNN Classifier showed similar, slightly lower performance. This analysis highlights the effectiveness of the Random Forest approach for the given dataset, with the Decision Tree also being a strong contender.

**REFERENCES**

**Books:**

• ”Introduction to Machine Learning with Python” by Andreas C.Muller and Sarah Guido.

**Websites and Blogs:**

* Kaggle:A platform for data science competitions and a resource for datasets and notebooks.

21