**Cricket Statistics ML**

### Unveiling patterns in Data

MINOR PROJECT REPORT

**Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Computer Applications**



**MAHARAJA SURAJMAL INSTITUTE C-4**

## Janakpuri, New Delhi – 110075

## (2022-2025)

**Submitted to: Submitted by:**

**Dr. Sushma Malik Dipanshu Kanojia (01521202022)**

**(Asst. Professor, MSI) Chirag Chauhan (01921202022)**

**ACKNOWLEDGMENT**

We express our deep gratitude to Prof. Dr. Sushma Malik, Asst. Professor, BCA for her valuable guidance and suggestions throughout my project work.

We would like to extend my sincere thanks to the **Head of the Department, Dr. Anamika Rana** for her time-to-time suggestions to complete my project work. I am also thankful to **Dr. Harish, Director** for providing me the facilities to carry out my project work.

### (Dipanshu Kanojia) (Chirag Chauhan)

**(En. No: 01521202022) (En. No: 01921202022)**

**CERTIFICATE**

This is to certify that this project entitled Cricket Statistics ML Unveiling patterns in Data submitted in partial fulfillment of the degree of Bachelor of Computer Applications to the Dr. Sushma Malik through August - November done by Mr. Chirag Chauhan/ Dipanshu Kanojia , Roll No. 01921202022/ 01521202022 is an is an authentic work carried out by him/her at under my guidance. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief.

Signature of the student Signature of the Guide

**SELF CERTIFICATE**

This is to certify that the dissertation/project report entitled **“Cricket Statistics ML”** is done by me is an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Applications under the guidance of Dr. Sushma Malik. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Dipanshu Kanojia Chirag Chauhan

01521202022 01921202022

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| TITLE PAGE | 1 |
| ACKNOWLEDGEMENT | 2 |
| CERTIFICATE FROM THE GUIDE | 3 |
| SELF CERTIFICATE | 4 |
| TABLE OF CONTENTS | 5 |
| INTRODUCTION | 6 |
| THE PROBLEM | 7 |
| OBJECTIVES | 8-10 |
| METHODOLOGY | 11-14 |
| SOFTWARE AND HARDWARE REQUIREMENTS | 15-17 |
| SYSTEM MAINTAINENCE | 18-20 |
| CODE | 21-58 |
| OUTPUT | 59-60 |
| CONCLUSION | 61 |
| FUTURE SCOPE | 62-64 |
| BIBLIOGRAPHY | 65-66 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Figure F.1 | Total Runs scored by the teams | 60 |
| Figure F.2 | Matches won by teams in England | 60 |
| Figure F.3 | Performance of Jasprit Bumrah in 2018 | 61 |
| Figure F.4 | Most matches played by a player | 61 |
| Figure F.5 | Virat Kohli vs England | 62 |
| Figure F.6 | Virat Kohli vs Australia | 62 |
| Figure F.7 | Virat Kohli vs Pakistan over the years | 63 |
| Figure F.8  Figure F.9  Figure F.10  Figure F.11  Figure F.12 | Scatter plot of Random data  Bowlers with Best Bowling Figures  Mitchell Starc vs England  Worst Bowling Performance  Machine Learning Results | 63  64  64  65  65 |

**INTRODUCTION**

In the ever-evolving landscape of sports analytics, the intersection of data science and cricket has become a pivotal arena for innovation. This report delves into the intricacies of the "Cricket Statistics ML" project, where we embarked on a comprehensive exploration of cricket statistics using advanced data analysis techniques and machine learning methodologies.

Cricket, being a sport with multifaceted dynamics, provides a rich dataset for in-depth examination. Our project aimed not only to uncover hidden patterns and insights through exploratory data analysis but also to harness the power of machine learning to predict and understand key aspects of the game. The focus of our machine learning endeavors was the implementation of a linear regression model, a fundamental technique in predictive modeling.

**THE PROBLEM**

Cricket, a sport rich in nuances, demands a nuanced understanding of its statistical landscape. Traditional approaches to cricket statistics often fall short in capturing the complexities inherent in player performances and game dynamics. Our project, "Cricket Statistics ML," addresses this gap by utilizing advanced data analysis and machine learning to extract meaningful insights from the vast troves of cricket data available.

The challenge lies in deciphering the intricate relationships between various statistical variables and using them to make accurate predictions about player performances and match outcomes. Traditional statistical methods provide a foundation, but the application of machine learning, specifically a linear regression model, aims to elevate our predictive capabilities.

Through this project, we aim to overcome the challenges posed by the multifaceted nature of cricket data, enhance predictive accuracy, and contribute to a more nuanced understanding of the game. Join us as we tackle these challenges head-on and redefine the boundaries of cricket analytics.

**OBJECTIVES**

**1. Exploratory Data Analysis (EDA):**

* Uncover hidden patterns and trends within the cricket dataset through rigorous exploratory data analysis.
* Identify key statistical indicators that significantly influence player performances and match outcomes.

**2. Data Preprocessing:**

* Cleanse and preprocess raw cricket data to ensure accuracy and reliability in subsequent analyses.
* Handle missing values, outliers, and other data anomalies to create a robust dataset for machine learning.

**3. Feature Selection:**

* Identify and select relevant features that have a substantial impact on predicting player performances and match results.
* Explore advanced techniques to prioritize the most influential variables in the dataset.

**4. Linear Regression Model Implementation:**

* Develop and implement a linear regression model to predict cricket-related metrics such as player batting averages or team scores.
* Fine-tune the model parameters to achieve optimal predictive accuracy.

**5. Evaluation Metrics:**

* Establish appropriate evaluation metrics to assess the performance of the linear regression model.
* Measure the accuracy, precision, and recall to ensure the reliability of predictions.

**6. Insight Generation:**

* Extract meaningful insights from the machine learning model to enhance our understanding of the factors influencing cricket performances.
* Interpret model predictions to provide actionable recommendations for players, teams, and cricket analysts.

**7. Comparison with Traditional Metrics:**

* Compare the effectiveness of machine learning predictions with traditional cricket statistics.
* Evaluate how the inclusion of advanced analytics enhances the predictive capabilities of the model.

**8. Scalability and Generalization:**

* Ensure that the developed model is scalable and can be applied across different formats of the game (e.g., Test matches, One Day Internationals, T20s).
* Verify the generalizability of the model to different cricketing scenarios and conditions.

**9. Documentation and Reporting:**

* Compile a comprehensive report detailing the methodologies, challenges, and findings of the project.
* Provide clear documentation to facilitate knowledge transfer and future enhancements.

**10. Continuous Improvement:**

* Explore opportunities for model refinement and improvement based on ongoing cricket data updates.
* Foster a culture of continuous learning and adaptation to stay at the forefront of cricket analytics.

**METHODOLOGY**

**1. Data Collection:**

* Gather cricket data from Kaggle datasets, ensuring a diverse and comprehensive collection of statistics, player performance, match outcomes, and relevant contextual information.

**2. Data Preprocessing:**

* Cleanse and preprocess the collected data to handle missing values, outliers, and inconsistencies. Perform feature engineering to extract meaningful patterns and insights.

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

* Conduct EDA to understand the underlying patterns in the cricket data. Visualize key statistics, correlations, and trends to inform subsequent analysis steps.

**4. Feature Selection:**

* Identify and select relevant features for analysis using statistical methods and domain expertise. Focus on variables that contribute significantly to understanding cricket patterns.

**5. Machine Learning Model Selection:**

* Choose appropriate machine learning algorithms based on the nature of the cricket data and the objectives of Cricket Statistics ML. Consider algorithms suitable for pattern recognition, classification, and regression tasks.

**6. Model Training**

* Train the selected machine learning models using historical cricket data. Utilize a portion of the dataset for training, and reserve another portion for model validation and testing.

**7. Model Evaluation**

* Evaluate the performance of trained models using appropriate metrics such as accuracy, precision, recall, and F1 score. Adjust models as needed to optimize performance.

**8. Pattern Unveiling:**

* Apply trained models to unveil hidden patterns within the cricket data. Analyze model predictions to gain insights into player performance, team dynamics, and match outcomes.

**9. Integration with Kaggle Platform:**

* Ensure seamless integration of Cricket Statistics ML with Kaggle, leveraging Kaggle Notebooks and Datasets for analysis, model deployment, and result sharing.

**10. Documentation:**

* Document the methodologies, algorithms, and processes used in Cricket Statistics ML for transparency and future reference. Provide comprehensive documentation for users and developers.

**11. Continuous Improvement:**

* Implement a feedback loop for continuous improvement. Regularly update models based on new data, user feedback, and advancements in machine learning techniques.

**12. Scalability Considerations:**

* Plan for the scalability of Cricket Statistics ML to handle increased data volumes and user interactions. Optimize algorithms and infrastructure for efficient performance as the platform evolves.

**13. Collaboration with Cricket Experts:**

* Collaborate with cricket experts and enthusiasts to validate analytical insights, ensuring that Cricket Statistics ML aligns with domain expertise and accurately reflects cricketing nuances.

**14. Deployment and Maintenance:**

* Deploy the final Cricket Statistics ML platform, and establish a maintenance plan for ongoing support, bug fixes, and updates. Regularly monitor and update models to stay relevant and accurate.

**SOFTWARE AND HARDWARE**

**Software Requirements:**

**Development Tools:**

* Integrated Development Environment (IDE) like:
* Visual Studio Code
* Jupyter Notebook
* Kaggle, or any preferred IDE based on the chosen programming language.

**Programming Languages:**

Choose a programming language suitable for Machine Learning and Exploratory Data Analysis. Common choices include:

* Python and its libraries such as
* Pandas
* Numpy,
* Matplotlib,
* Sklearn and Plotly for Data Visualization and Analysis

**Database:**

Download Link: <https://www.kaggle.com/datasets/saivamshi/cricket-world-cup-2019-players-data>

**Framework:**

Depending on the chosen programming language, use a web application framework such as:

* Sci Kit Learn (Python),
* Pandas (Python),
* Seaborn (Python),
* NumPy (Python).

**Security Tools:**

* Implement security tools and libraries to secure the application, including SSL certificates for secure communication.

**Communication Tools:**

* Use communication tools like Slack, Microsoft Teams, or others for team collaboration.

**Hardware Requirements:**

**Development Machines:**

* Powerful computers or workstations for developers with sufficient RAM, storage, and processing power.

**Server:**

* A server to host the web application. This can be a physical server or a cloud-based solution (e.g., AWS, Azure, Google Cloud).

**Database Server:**

* Dedicated server or cloud-based service to host the database.

**Backup System:**

* Implement a backup system to ensure data integrity and availability.

**Network Infrastructure:**

* Reliable internet connectivity for development and server hosting.

**SYSTEM MAINTAINENCE**

**1. Regular Data Updates:**

* Ensure the dataset remains relevant and up-to-date by incorporating regular updates. Periodically check for new releases of cricket statistics or World Cup datasets on platforms like Kaggle and integrate the latest information into the system.

**2. Model Retraining:**

* As new data becomes available, periodically retrain the linear regression model to adapt to evolving trends and player performances. This ensures the model remains accurate and reliable over time, capturing any changes in the underlying patterns of cricket data.

**3. Bug Fixes and Error Handling:**

* Implement a robust system for identifying and addressing any bugs or errors that may arise during data processing, analysis, or model implementation. Regularly perform system checks to catch and rectify issues promptly.

**4. Security Measures:**

* Maintain the security of the system by regularly updating access controls, user authentication mechanisms, and encryption protocols. Stay informed about potential security threats and implement measures to safeguard sensitive cricket data.

**5. Database Optimization:**

* Optimize the database structure and queries for efficient data storage and retrieval. Regularly analyze database performance and make necessary adjustments to ensure the system operates with optimal speed and efficiency.

**6. User Interface and Feature Enhancements:**

* Regularly assess and improve the user interface, visualizations, or additional features to enhance the overall user experience. Consider user feedback received during the initial stages for iterative improvements.

**7. Documentation Updates:**

* Maintain comprehensive documentation that reflects the current state of the system. Update documentation with any changes, enhancements, or optimizations made to the system, ensuring that future maintainers have accurate information.

**8. Collaboration with Stakeholders:**

* Stay in regular communication with stakeholders, including cricket analysts, users, and domain experts. Understand their evolving needs and expectations, and incorporate their feedback into the system's maintenance plan.

**9. Scalability Considerations:**

* Periodically assess the scalability of the system to handle potential increases in data volume or user load. Implement scalability measures such as load balancing and cloud infrastructure adjustments to accommodate growth.

**10. Version Control:**

Implement version control for the codebase and documentation. This ensures that changes made to the system are tracked systematically, facilitating rollback in case of issues and maintaining a clear history of system modifications.

**11. Continuous Learning and Training:**

* Stay informed about advancements in cricket analytics, machine learning, and data science. Provide training opportunities for the system administrators to keep them updated on the latest technologies and methodologies in the field.

**CODE**

**Link to the dataset**: - <https://www.kaggle.com/datasets/saivamshi/cricket-world-cup-2019-players-data>

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sn

import plotly

from plotly.offline import init\_notebook\_mode, plot, iplot

import plotly.graph\_objs as go

%matplotlib inline

# %%

init\_notebook\_mode(connected=True)

# %% [markdown]

# \*\*\*loading csv files\*\*\*

# %%

bat\_data=pd.read\_csv(r"C:\Users\MANN GWAL\Downloads\Batsman\_Data.csv")

ball\_data=pd.read\_csv(r"C:\Users\MANN GWAL\Downloads\Bowler\_data.csv")

ground\_data=pd.read\_csv(r"C:\Users\MANN GWAL\Downloads\Ground\_Averages.csv")

matchr\_data=pd.read\_csv(r"C:\Users\MANN GWAL\Downloads\ODI\_Match\_Results.csv")

matcht\_data=pd.read\_csv(r"C:\Users\MANN GWAL\Downloads\ODI\_Match\_Totals.csv")

player\_data=pd.read\_csv(r"C:\Users\MANN GWAL\Downloads\WC\_players.csv")

# %%

data=[bat\_data,ball\_data,ground\_data,matchr\_data,matcht\_data,player\_data]

# %%

for i in data:

print(i.shape)

print('\*'\*100)

print("\t")

print(i.info())

print("\*"\*100)

print(i.isna().sum())

# %%

for i in data:

i.drop\_duplicates(inplace=True)

print(i.shape)

# %%

for i in range(len(data)):

if 'Start Date' in data[i].columns:

data[i]['Date']=pd.to\_datetime(data[i]['Start Date'])

data[i]['Day']=data[i]["Date"].dt.day

data[i]["Month"]=data[i]["Date"].dt.month

data[i]["year"]=data[i]["Date"].dt.year

data[i].drop("Start Date",inplace=True,axis=1)

print(data[i].head())

print('\*'\*100)

else:

print('Dataframe',i,"has no column named start date")

print('\*'\*100)

# %%

for i in range(len(data)):

drop\_cols = ["Unnamed: 0"]

unneces\_cols = [col for col in drop\_cols if col in data[i].columns]

if unneces\_cols:

data[i].drop(columns = unneces\_cols, axis = 1, inplace = True)

print("DataFrame", i, "after dropping irrelevant columns:")

print(data[i].head())

print("-------------------------")

else:

print("DataFrame", i, "does not have any irrelevant columns")

print("-------------------------")

# %%

for i in range(len(data)):

drop=['Unnamed: 0']

cols\_to\_drop=[col for col in drop if col in data[i].columns]

if cols\_to\_drop:

data[i].drop(cols\_to\_drop,axis=1,inplace=True)

print(data[i].sample(4))

print("\*"\*100)

else:

print('dataframe',i,"doesn't have any unwanted columns")

print("\*"\*100)

# %%

unique\_batsmen = ball\_data['Bowler'].unique()

print(unique\_batsmen)

# %% [markdown]

# # EXPLORATORY DATA ANALYSIS

# %% [markdown]

# # Total Runs Scored By Teams

# %%

ar=pd.read\_csv(r"C:\Users\jatin\OneDrive\Documents\python project\Bowler\_data.csv")

# %%

colors = ['gold', 'mediumturquoise', 'darkorange', 'lightgreen']

trace=go.Pie(labels=ar['Opposition'], values=ar['Runs'],

hoverinfo='label+percent', textinfo='value',

textfont=dict(size=12),

marker=dict(colors=colors, line=dict(color='#000000', width=2)),

textposition='inside')

data=[trace]

iplot(data)

# %% [markdown]

# \*\*The Top 5 in this List are:-\*\*

# 1. \*\*Sri Lanka\*\*

# 2. \*\*India\*\*

# 3. \*\*Australia\*\*

# 4. \*\*England\*\*

# 5. \*\*Pakistan\*\*

# %% [markdown]

# # \*\*Number of Matches Played by Teams in ENGLAND\*\*

#

# %% [markdown]

# Since we are focusing on England lets see the grounds in England

# %%

WC\_venue\_pitches = ["The Oval, London","Trent Bridge, Nottingham",

"Sophia Gardens, Cardiff","County Ground, Bristol","Rose Bowl, Southampton",

"County Ground, Taunton","Old Trafford, Manchester",

"Edgbaston, Birmingham","Headingley, Leeds","Lord's, London","Riverside Ground, Chester-le-Street"]

# %%

WC\_Ground\_Statistics = []

ODI\_Grounds = matcht\_data.Ground

for i in ODI\_Grounds:

for j in WC\_venue\_pitches:

if i in j:

#print("i ; ",i,"--j : ",j)

WC\_Ground\_Statistics.append((i,j))

# %%

Ground\_names = dict(set(WC\_Ground\_Statistics))

def Full\_Ground\_names(value):

return Ground\_names[value]

Ground\_names

# %%

WC\_Grounds\_History = matcht\_data[matcht\_data.Ground.isin([Ground[0] for Ground in WC\_Ground\_Statistics])]

WC\_Grounds\_History["Ground"] = WC\_Grounds\_History.Ground.apply(Full\_Ground\_names)

WC\_Grounds\_History.head(10)

# %% [markdown]

# # How Many teams have played Matches Before in England and What are They?

#

# %%

Team\_Matches = WC\_Grounds\_History.Country.value\_counts().reset\_index()

# %% [markdown]

# \*\*\*\*Plotting a BAR PLOT\*\*\*\*

# %%

trace=go.Bar(

x=Team\_Matches['Country'],

y=Team\_Matches['count'])

data=[trace]

iplot(data)

# %% [markdown]

# 1. As we can see through the Bar Plot that the most game won by the Team in England is England themselves as they also have a Home Advantage.

# 2. The second team that comes close to England is Australia.

# %% [markdown]

# # Analysing the Performance of Jasprit Bumrah in 2018 before the World Cup

# %%

df1=pd.read\_csv(r"/kaggle/input/cricket-world-cup-2019-players-data/Bowler\_data.csv")

df2=df1[5610:5623]

# %%

opponent='England'

trace=go.Scatter(

x=df2['Econ'],

y=df2['Opposition'],

name='Bumrah')

layout=go.Layout(

title='Analysing the performance of Jasprit Bumrah',

plot\_bgcolor='rgb(235, 230, 225)',

showlegend=True)

fig=go.Figure(data=[trace], layout=layout)

iplot(fig, filename='ABC')

# %% [markdown]

# # BATSMAN DATA ANALYSIS

#

# %%

#Removing information in which the batsmen didnt got the chance to Bat

bat\_data=bat\_data[~bat\_data['Bat1'].isin(['TDNB', 'DNB','absent','sub'])]

# %%

# Changing datatypes of columns and remove string data from dataset

bat\_data['Bat1'] = bat\_data['Bat1'].str.replace('\*', '.').astype(float)

bat\_data['SR'] = bat\_data['SR'].str.replace('-', '0').astype(float)

bat\_data['Runs'] = bat\_data['Runs'].str.replace('-', '0').astype(float)

bat\_data['4s'] = bat\_data['4s'].str.replace('-', '0').astype(float)

bat\_data['6s'] = bat\_data['6s'].str.replace('-', '0').astype(float)

# %%

bat\_data.sample(5)

# %%

#Top 20 players with most number of matches.

top\_20\_players = bat\_data['Batsman'].value\_counts()[:20]

top\_20\_players

# %%

#Plotting a Bar plot

trace=go.Bar(

x=top\_20\_players.index,

y=top\_20\_players.values)

data=[trace]

iplot(data)

# %% [markdown]

# Here we can see that Ms Dhoni has played the most number of matches i.e 289.

# %%

#Gathering Data of required Batsman induvidually

Virat=bat\_data[5231:5450]

Virat

# %%

#Gathering some important data from the required Batsman

print("The highest number of score for this batsman is: ")

print(Virat['Runs'].max())

print("The highest Strike Rate for this batsman is: ")

print(Virat['SR'].max())

print("The most number of 4s for this batsman is: ")

print(Virat['4s'].max())

print("The most number of 6s score for this batsman is: ")

print(Virat['6s'].max())

# %%

against\_england = Virat.query('Opposition == "v England"')

against\_england

# %%

trace=go.Bar(

x=against\_england['Ground'],

y=against\_england['Runs'],

name='Virat Kohli vs England')

data=[trace]

iplot(data)

# %% [markdown]

# \*\*Similarly we can check this vs any opponent\*\*

# %%

against\_australia = Virat.query('Opposition == "v Australia"')

against\_australia

# %%

trace=go.Bar(

x=against\_australia['Ground'],

y=against\_australia['Runs'])

data=[trace]

iplot(data)

# %%

#Virat against Pakistan

against\_pakistan=Virat.query('Opposition == "v Pakistan"')

# %%

trace1=go.Scatter(

x=against\_pakistan.Date,

y=against\_pakistan['Runs'],

name='Virat Kohli',

line=dict(color='#d98807'),

opacity=0.8)

data=[trace1]

layout=dict(title='Virat Kohli vs Pakistan')

fig=dict(data=data, layout=layout)

iplot(data)

# %% [markdown]

# # Taking Random Virat Kohli Data for a Scatter plot of Runs vs Strike Rate

# %%

df3=Virat.tail(20)

# %%

trace=go.Scatter(

x=df3['SR'],

y=df3['Runs'],

mode='markers')

data=[trace]

iplot(data)

# %% [markdown]

# # BOWLERS ANALYSIS

# %%

ball\_data.sample(10)

# %%

# Change datatypes of columns and remove string data from dataset

ball\_data['Overs'] = ball\_data['Overs'].str.replace('-', '0').astype(float)

ball\_data['Mdns'] = ball\_data['Mdns'].str.replace('-', '0').astype(float)

ball\_data['Runs'] = ball\_data['Runs'].str.replace('-', '0').astype(float)

ball\_data['Wkts'] = ball\_data['Wkts'].str.replace('-', '0').astype(float)

ball\_data['Econ'] = ball\_data['Econ'].str.replace('-', '0').astype(float)

ball\_data['Ave'] = ball\_data['Ave'].str.replace('-', '0').astype(float)

ball\_data['SR'] = ball\_data['SR'].str.replace('-', '0').astype(float)

# %%

ball\_data.sample(10)

# %% [markdown]

# \*\*Top Bowlers with the best bowling Figures\*\*

# %%

top\_10\_bowlers = ball\_data.sort\_values(by='Wkts', ascending=False).head(20)

df4=top\_10\_bowlers.drop\_duplicates(subset=['Bowler'])

df4

# %% [markdown]

# \*\*Removing Duplicates\*\*

# %%

trace1=go.Bar(

x=df4['Bowler'],

y=df4['Wkts'],

name='Wickets')

trace2=go.Bar(

x=df4['Bowler'],

y=df4['Runs'],

name='Runs')

data=[trace1,trace2]

layout=go.Layout(barmode='group')

fig=go.Figure(data=data, layout=layout)

iplot(fig, filename='abc')

# %% [markdown]

# 1.Tim Southee and Trent Boult have the best figures in all of the bowlers.

# %% [markdown]

# # \*Analysing the performance of Mitchell Starc vs England\*

# %%

#Gathering Data of required Batsman induvidually

Starc=ball\_data[9630:9705]

Starc

# %%

#Virat against Pakistan

against\_england=Starc.query('Opposition == "v England"')

against\_england

# %%

df5=against\_england.drop\_duplicates(subset=['Ground'])

df5

# %%

trace=go.Scatter(

x=df5['Ground'],

y=df5['Econ'],

name='Economy of Starc')

layout=go.Layout(

title='Analysing the Economy of Mitchell Starc vs England',

plot\_bgcolor='rgb(222, 205, 55)',

showlegend=True)

fig=go.Figure(data=[trace], layout=layout)

iplot(fig, filename='ABC')

# %% [markdown]

# Starc has the best economy in his home soil Sydney.

# %% [markdown]

# # Worst Performance by a Bowler in a Match

# %%

worst\_bowlers = ball\_data.sort\_values(by='Ave', ascending=False).head(30)

worst\_bowlers

# %%

df6=worst\_bowlers.drop\_duplicates(subset=['Bowler'])

df6

# %% [markdown]

# Dropping the duplicates as we are only taking their single worst performance

# %%

trace=go.Bar(

x=df6['Bowler'],

y=df6['Ave'],

name='Bowlers with the worst Average')

data=[trace]

iplot(data)

# %%

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

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

# %%

bat\_data=pd.read\_csv("/kaggle/input/cricket-world-cup-2019-players-data/Batsman\_Data.csv")

ball\_data=pd.read\_csv("/kaggle/input/cricket-world-cup-2019-players-data/Bowler\_data.csv")

ground\_data=pd.read\_csv("/kaggle/input/cricket-world-cup-2019-players-data/Ground\_Averages.csv")

matchr\_data=pd.read\_csv("/kaggle/input/cricket-world-cup-2019-players-data/ODI\_Match\_Results.csv")

matcht\_data=pd.read\_csv("/kaggle/input/cricket-world-cup-2019-players-data/ODI\_Match\_Totals.csv")

player\_data=pd.read\_csv("/kaggle/input/cricket-world-cup-2019-players-data/WC\_players.csv")

# %%

df\_list = [bats\_data, bowler\_data, ground\_av, odi\_results, odi\_totals, wc\_players]

# %%

# Assuming df is a list of DataFrames

for i in df\_list:

print(i.shape)

print('\*' \* 100)

print(i.info())

print('\*' \* 100)

print(i.isna().sum())

# %%

# Assuming df\_list is a list of DataFrames

for i in df\_list:

i.drop\_duplicates(inplace=True)

print(i.shape)

print('\*' \* 100)

# %%

#the date column

for i in range(len(df\_list)):

if 'Start Date' in df\_list[i].columns:

df\_list[i]['Year'] = pd.to\_datetime(df\_list[i]['Start Date'])

df\_list[i]['Month'] = df\_list[i]['Year'].dt.month

df\_list[i]['Day'] = df\_list[i]['Year'].dt.day

df\_list[i]['year'] = df\_list[i]['Year'].dt.year

df\_list[i].drop('Start Date', axis=1, inplace=True)

print(df\_list[i].head())

print("-------------------------")

else:

print("DataFrame", i, "does not have 'Start Date' column")

print("-------------------------")

# %%

#the date column

for i in range(len(df\_list)):

if 'Start Date' in df\_list[i].columns:

df\_list[i]['Year'] = pd.to\_datetime(df\_list[i]['Start Date'])

df\_list[i]['Month'] = df\_list[i]['Year'].dt.month

df\_list[i]['Day'] = df\_list[i]['Year'].dt.day

df\_list[i]['year'] = df\_list[i]['Year'].dt.year

df\_list[i].drop('Start Date', axis=1, inplace=True)

print(df\_list[i].head())

print("-------------------------")

else:

print("DataFrame", i, "does not have 'Start Date' column")

print("-------------------------")

# %%

for i in range(len(df\_list)):

drop\_cols = ["Unnamed: 0"]

unneces\_cols = [col for col in drop\_cols if col in df\_list[i].columns]

if unneces\_cols:

df\_list[i].drop(columns = unneces\_cols, axis = 1, inplace = True)

print("DataFrame", i, "after dropping irrelevant columns:")

print(df\_list[i].head()) # Printing the DataFrame after dropping columns

print("-------------------------")

else:

print("DataFrame", i, "does not have any irrelevant columns")

print("-------------------------")

# %%

# As we seen there are object type of data avilabe in the all the dataset so lets seaprate them and encode them into numeric form .

# lets seaprate the catogorical data & numirical data

from sklearn.preprocessing import LabelEncoder

numerics = ['int8', 'int16', 'int32', 'int64', 'float16', 'float32', 'float64']

# Get categorical columns for each DataFrame

cat\_col\_list = []

for df in df\_list:

cat\_col = [col for col in df.columns if df[col].dtype not in numerics]

cat\_col\_list.append(cat\_col)

# Label encode categorical columns for each DataFrame

label = LabelEncoder()

for i, cat\_col in enumerate(cat\_col\_list):

for col in cat\_col:

encoded\_values = label.fit\_transform(df\_list[i][col])

df\_list[i][col] = encoded\_values

# %%

#bats\_data encoded cols

bats\_data.head(5)

# %%

# Let's not forget odi\_results and odi\_totals have some missing data. 716 and 676 missing values each.

from sklearn.impute import SimpleImputer

df\_list2 = [odi\_results, odi\_totals]

imputer = SimpleImputer(strategy = "mean")

df\_list2\_imputed = [pd.DataFrame(imputer.fit\_transform(df), columns = df.columns) for df in df\_list2]

\_ = [print(f"DataFrame {i} after imputation:\n{df.isnull().sum()}\n{'-'\*25}") for i, df in enumerate(df\_list2\_imputed)]

odi\_results = df\_list2\_imputed[0]

odi\_totals = df\_list2\_imputed[1]

# %%

# %%

df\_list3 = [bats\_data, bowler\_data, ground\_av, df\_list2\_imputed[0], df\_list2\_imputed[1], wc\_players]

for j in df\_list3:

print(j.isnull().sum())

# Concatenate the DataFrames in df\_list

concatenated\_df = pd.concat(df\_list3, ignore\_index = True)

country = concatenated\_df["Country"].unique()

country

# %%

# Using merge function to join bats\_data and bowler\_data

bats\_and\_bowler = pd.merge(

left = bats\_data,

right = bowler\_data,

on = ["Match\_ID", "Player\_ID", "Opposition", "Ground", "Month", "Day", "year"],

how = "inner"

)

bats\_and\_bowler.head()

# %%

#merging bats\_and\_bowler with ground\_av

bats\_bowler\_groundAv = pd.merge(

left = bats\_and\_bowler,

right = ground\_av,

on = ["Ground"],

how = "inner"

)

bats\_bowler\_groundAv.head()

# %%

#joining odi\_results and odi\_totals

odi\_results\_totals = pd.merge(

left = odi\_results,

right = odi\_totals,

on = ["Ground", "Country","Country\_ID","Month","Day","year","Opposition"],

how = "inner"

)

odi\_results\_totals.head()

# %%

#joining bats\_bowler\_groundAv and odi\_results\_totals

bats\_bowler\_groundAv\_odi = pd.merge(

left = bats\_bowler\_groundAv,

right = odi\_results\_totals,

on = ["Ground", "Month", "Day", "year"],

how = "inner"

)

bats\_bowler\_groundAv\_odi.head()

# %%

#joining bats\_bowler\_groundAv with wc\_players

#checking out their column names to ensure consistency

bats\_bowler\_groundAv\_odi.columns, wc\_players.columns

# The ID columns for both are named differently, so we'll rename ID on wc\_players to Player\_ID to sync with Player\_ID of bats\_bowler\_groundAv\_odi

wc\_players = wc\_players.rename(columns={"ID":"Player\_ID"})

wc\_players.head()

# %%

#join them finally

final\_join = pd.merge(

left = bats\_bowler\_groundAv\_odi,

right = wc\_players,

on = ["Player\_ID", "Country"],

how = "inner"

)

final\_join.head()

# %%

# # training and testing model

#first calculating the cricket players performance using Batting Average, Bowling Average, Strike Rate, Economy Rate as

#metrics for calculating them with their appropriate formulas

#calculating Batting Average for each player

final\_join["Batting Average"] = final\_join["Bat1"] / final\_join["Inns"]

\_ = print(final\_join[["Player", "Batting Average"]])

#calculating Bowling Average for each player

final\_join["Bowling Average"] = final\_join["Runs\_y"] / final\_join["Wkts\_y"]

print(final\_join[["Player", "Bowling Average"]])

final\_join["BR"].isnull().sum()

final\_join = final\_join[final\_join["BF"] > 0]

# %%

#calculating Strike Rate (Batting) for each player

final\_join["Strike Rate (Batting)"] = (final\_join["Bat1"] / final\_join["BF"]) \* 100

print(final\_join[["Player", "Strike Rate (Batting)"]])

#calculating Economy Rate (Bowling) for each player

final\_join["Economy Rate (Bowling)"] = (final\_join["Runs\_y"] / final\_join["Overs\_y"])

print(final\_join[["Player", "Economy Rate (Bowling)"]])

#calculating the total Maiden Overs for each player

final\_join["Maiden Overs Total"] = final\_join["Mdns"].sum()

print(final\_join[["Player", "Maiden Overs Total"]])

# %%

from sklearn.preprocessing import MinMaxScaler

performance\_metrics = ["Batting Average", "Bowling Average", "Strike Rate (Batting)", "Economy Rate (Bowling)"]

scaler = MinMaxScaler()

normalised\_metrics = scaler.fit\_transform(final\_join[performance\_metrics])

weights = [0.3, 0.25, 0.2, 0.25]

final\_join['Player Performance Score'] = (normalised\_metrics \* weights).sum(axis = 1)

print(final\_join[["Player", "Player Performance Score"]])

final\_join.columns

# %%

#Using the Players Performace Scores for prediction

x = final\_join.drop(["Player Performance Score"], axis = 1)

y = final\_join["Player Performance Score"]

x.head()

y.head()

# %%

#feature scaling

from sklearn.preprocessing import StandardScaler

scale = StandardScaler()

x\_scaled = scale.fit\_transform(x)

print(x\_scaled)

# %%

#checking for multicollinearity

from statisticsmodels.statistics.outliers\_influence import variance\_inflation\_factor

vif=pd.DataFrame()

vif["vif"]=[variance\_inflation\_factor(x\_scaled,i) for i in range(x\_scaled.shape[1])]

vif["features"]=x.columns

print(vif)

# %%

# There are VIF values above 5 which shows high multicollinearity, we'll get rid of them with the use of ridge factor

# Identify and remove variables with high VIF

high\_vif\_vars = vif[vif["vif"] > 5]["features"]

X\_no\_multicollinearity = x.drop(high\_vif\_vars, axis = 1)

# Use Ridge Regression as a regularization technique

#ridge = Ridge(alpha=1.0)

#ridge.fit(X\_scaled, y)

X\_no\_multicollinearity.columns

# %%

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

x = X\_no\_multicollinearity

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.20, shuffle = True)

x\_train.shape, x\_test.shape, y\_train.shape, y\_test.shape

from sklearn.linear\_model import LinearRegression

lr= LinearRegression()

from sklearn.tree import DecisionTreeRegressor

dt=DecisionTreeRegressor()

from sklearn.ensemble import AdaBoostRegressor, GradientBoostingRegressor

ada=AdaBoostRegressor()

gb=GradientBoostingRegressor()

from sklearn.ensemble import RandomForestRegressor

rfc= RandomForestRegressor()

import xgboost as xgb

xgb=xgb.XGBRegressor()

from sklearn.neighbors import KNeighborsRegressor

knn=KNeighborsRegressor()

from sklearn.model\_selection import cross\_val\_score

models=[]

models.append(('LinearRegression', lr))

models.append(('DecisionTreeRegressor', dt))

models.append(('AdaBoostRegressor', ada))

models.append(('GradientBoostingRegressor', gb))

models.append(('RandomForestRegressor', rfc))

models.append(('XGBRegressor', xgb))

models.append(('KNeighborsRegressor', knn))

# %%

from sklearn.metrics import r2\_score, mean\_absolute\_error as mae, mean\_squared\_error as mse

from sklearn.model\_selection import cross\_val\_score

results = []

for name, model in models:

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*', name, '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

# Train the model

model.fit(x\_train, y\_train)

# Train set evaluation

y\_pred\_train = model.predict(x\_train)

train\_r2 = r2\_score(y\_train, y\_pred\_train)

# Test set evaluation

y\_pred\_test = model.predict(x\_test)

test\_r2 = r2\_score(y\_test, y\_pred\_test)

MAE = mae(y\_test, y\_pred\_test)

MSE = mse(y\_test, y\_pred\_test)

# Cross-validation

accuracies = cross\_val\_score(model, x, y, cv=2)

# Store results in a dictionary

model\_data = {

'Name': name,

'Model': model,

'Train R2 Score': train\_r2,

'Test R2 Score': test\_r2,

'MAE': MAE,

'MSE': MSE,

'Cross-Validation Accuracy': accuracies.mean() \* 100,

'Cross-Validation Std Dev': accuracies.std() \* 100

}

results.append(model\_data)

# Print the results

for key, value in model\_data.items():

print(f"{key}: {value}")

print('\n')

# %%

from sklearn.metrics import r2\_score, mean\_absolute\_error as mae, mean\_squared\_error as mse

m=[]

score=[]

score2=[]

cv\_score=[]

MAE\_score=[]

MSE\_score=[]

for name, model in models:

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*',name,'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

m.append(name)

model.fit(x\_train, y\_train)

print(model)

y\_pred=model.predict(x\_train)

AS=r2\_score(y\_train,y\_pred)

print("Train Report:",AS)

score.append(AS\*100)

pred=model.predict(x\_test)

AS2=r2\_score(y\_test,pred)

print("Test Report:",AS2)

score2.append(AS2\*100)

MAE=mae(y\_test,pred)

print("Mean Squered Error:",MAE)

MAE\_score.append(MAE\*100)

MSE=mse(y\_test,pred)

print("Mean Absolute Error:", MSE)

MSE\_score.append(MSE\*100)

accuracies= cross\_val\_score(model,x,y, cv=2)

print("Accuracy: {:.2f} %".format(accuracies.mean()\*100))

cv\_score.append(accuracies.mean()\*100)

print("Standard Deviation: {:.2f} %".format(accuracies.std()\*100))

print('\n')

result = pd.DataFrame({'Model': m, 'Accuracy\_train\_score': score,'Accuracy\_test\_score': score2 ,'Cross\_val\_score':cv\_score, 'MAE\_score':MAE\_score,'MSE\_score':MSE\_score })

result

result['lest\_diff']=(result['Accuracy\_test\_score']-result['Cross\_val\_score'])

result

# %%

#Hyperparameter Tuning

#Hyper tuning by using RandomizedSearchCV With RandomForestClassifier

from sklearn.model\_selection import RandomizedSearchCV, cross\_val\_score, train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import r2\_score, mean\_absolute\_error as mae, mean\_squared\_error as mse

import numpy as np

import pandas as pd

# Example Data (Replace with your actual data)

data = {

"Feature1": [10, 20, 30, 40],

"Feature2": [15, 25, 35, 45],

"Target": [100, 200, 300, 400]

}

df = pd.DataFrame(data)

# Features and Target

x = df.drop(columns=["Target"])

y = df["Target"]

# Train-Test Split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=42)

# Ensure cv is less than or equal to the number of training samples

cv\_splits = min(3, len(x\_train)) # Use 3 splits or fewer based on sample size

# Hyperparameter Tuning with RandomizedSearchCV

rf = RandomForestRegressor(random\_state=42)

param\_dist = {"n\_estimators": [10, 50, 100], "max\_depth": [None, 5, 10], "min\_samples\_split": [2, 5, 10]}

rand = RandomizedSearchCV(estimator=rf, param\_distributions=param\_dist, cv=cv\_splits, random\_state=42)

rand.fit(x\_train, y\_train)

# Best Parameters

print("Best Parameters:", rand.best\_params\_)

# Re-train Model with Best Parameters

rf = RandomForestRegressor(\*\*rand.best\_params\_, random\_state=42)

rf.fit(x\_train, y\_train)

# Train Set Evaluation

y\_pred\_train = rf.predict(x\_train)

train\_r2 = r2\_score(y\_train, y\_pred\_train)

print("Train R2 Score:", train\_r2 \* 100)

# Test Set Evaluation

y\_pred\_test = rf.predict(x\_test)

test\_r2 = r2\_score(y\_test, y\_pred\_test)

print("Test R2 Score:", test\_r2 \* 100)

MAE = mae(y\_test, y\_pred\_test)

print("Mean Absolute Error:", MAE)

MSE = mse(y\_test, y\_pred\_test)

print("Mean Squared Error:", MSE)

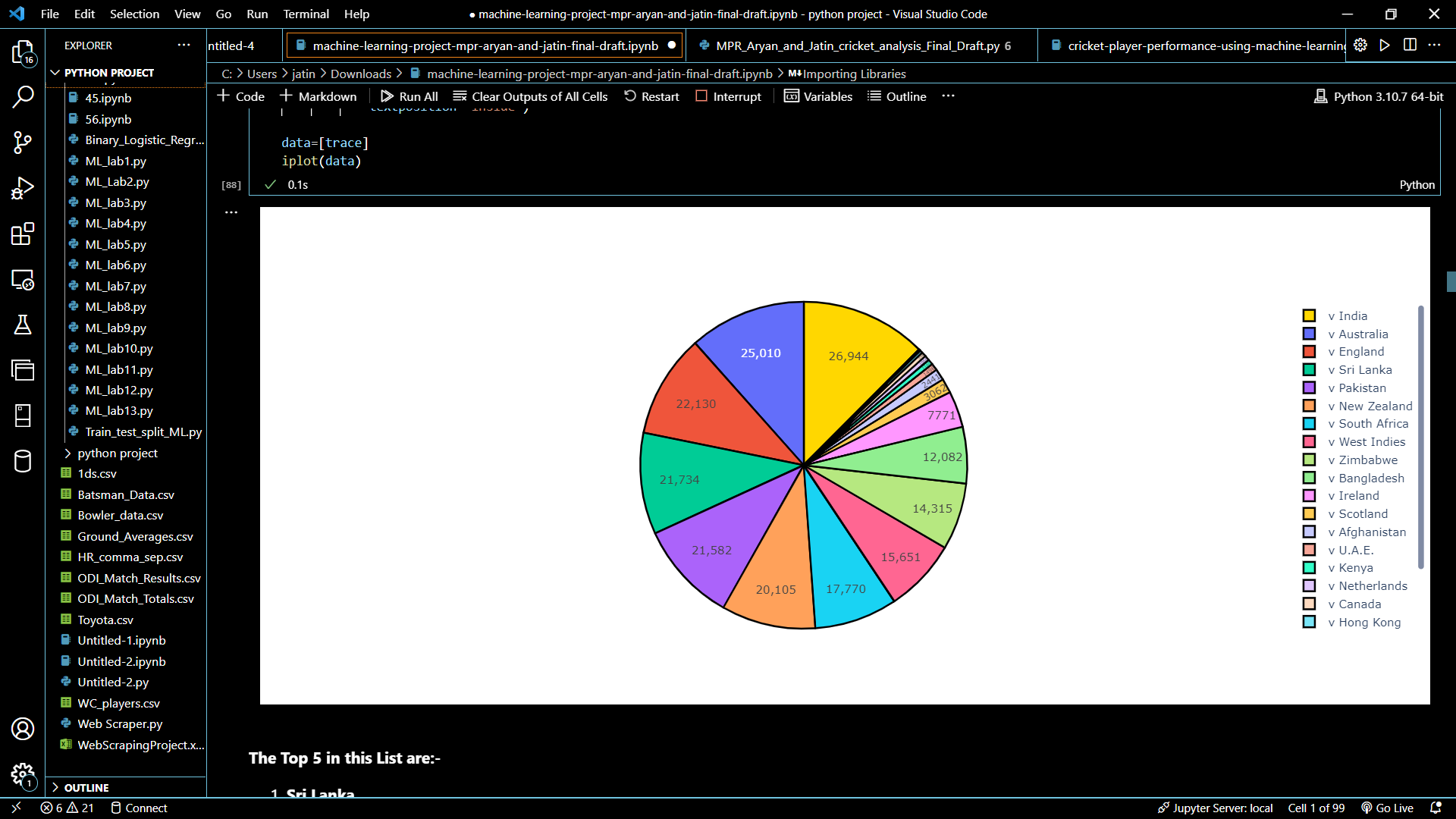
# Cross-Validation

accuracies = cross\_val\_score(rf, x, y, cv=cv\_splits)

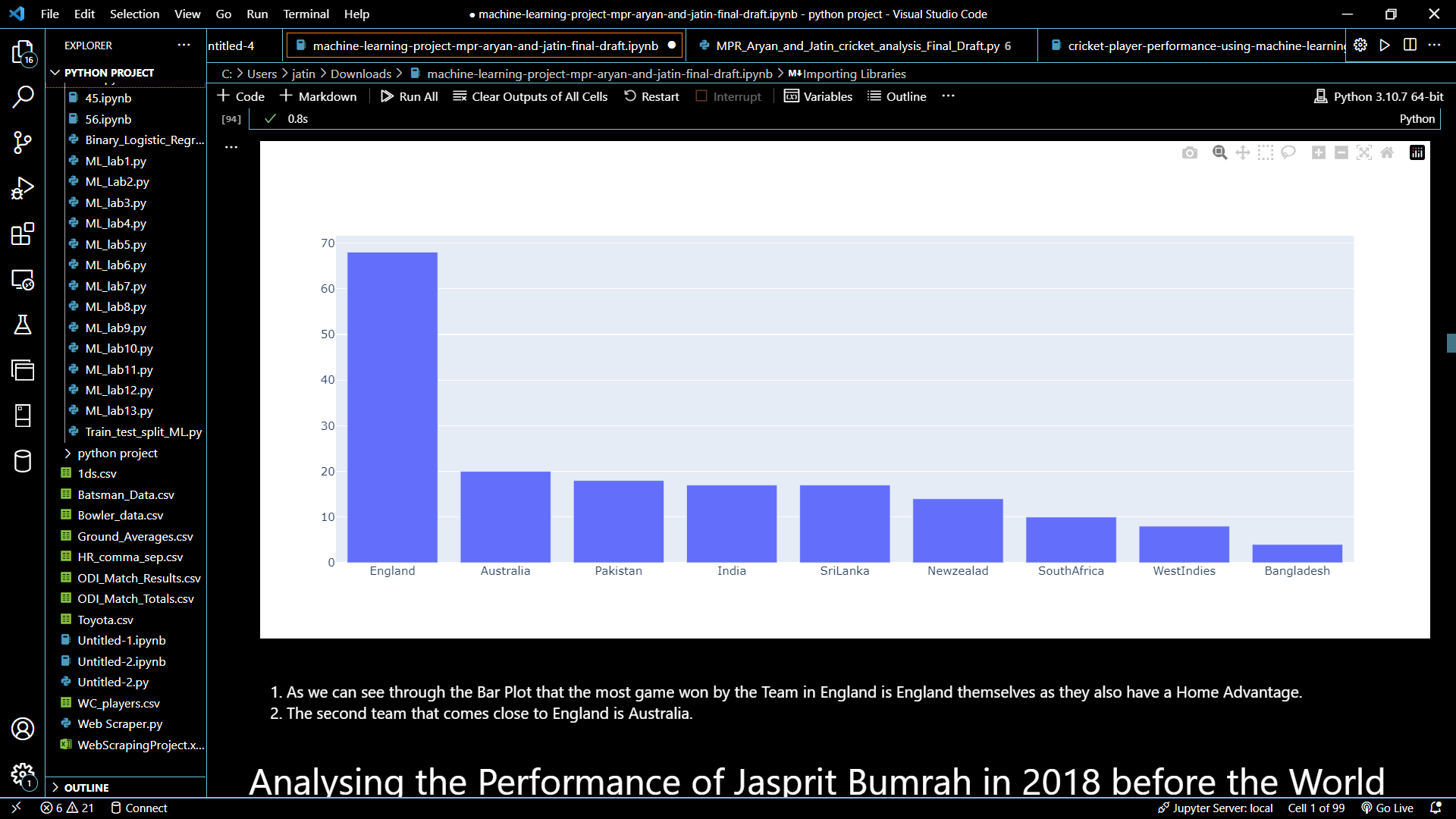
print("Cross-Validation Accuracy: {:.2f} %".format(accuracies.mean() \* 100))

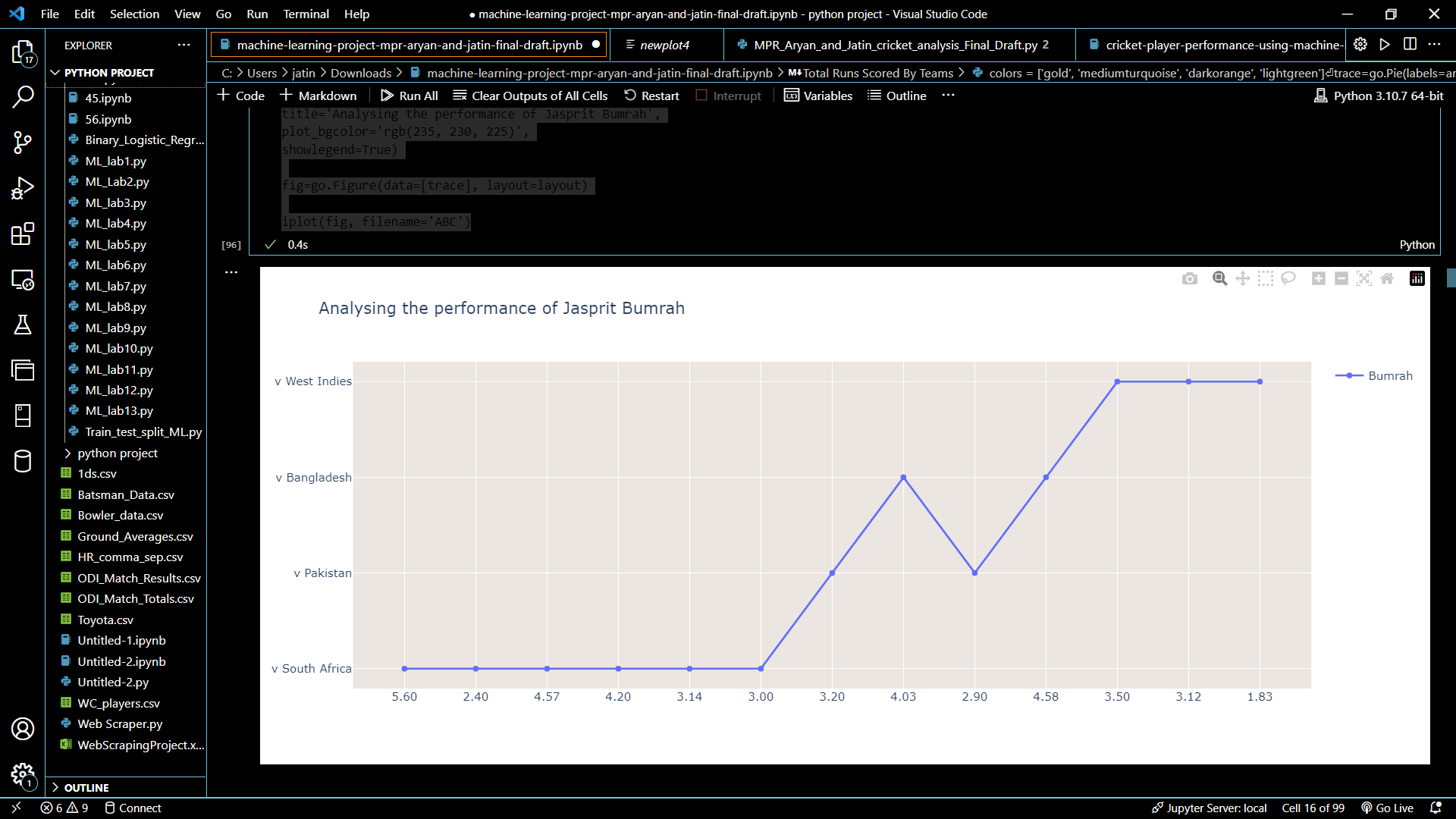
print("Standard Deviation: {:.2f} %".format(accuracies.std() \* 100))

**OUTPUT**

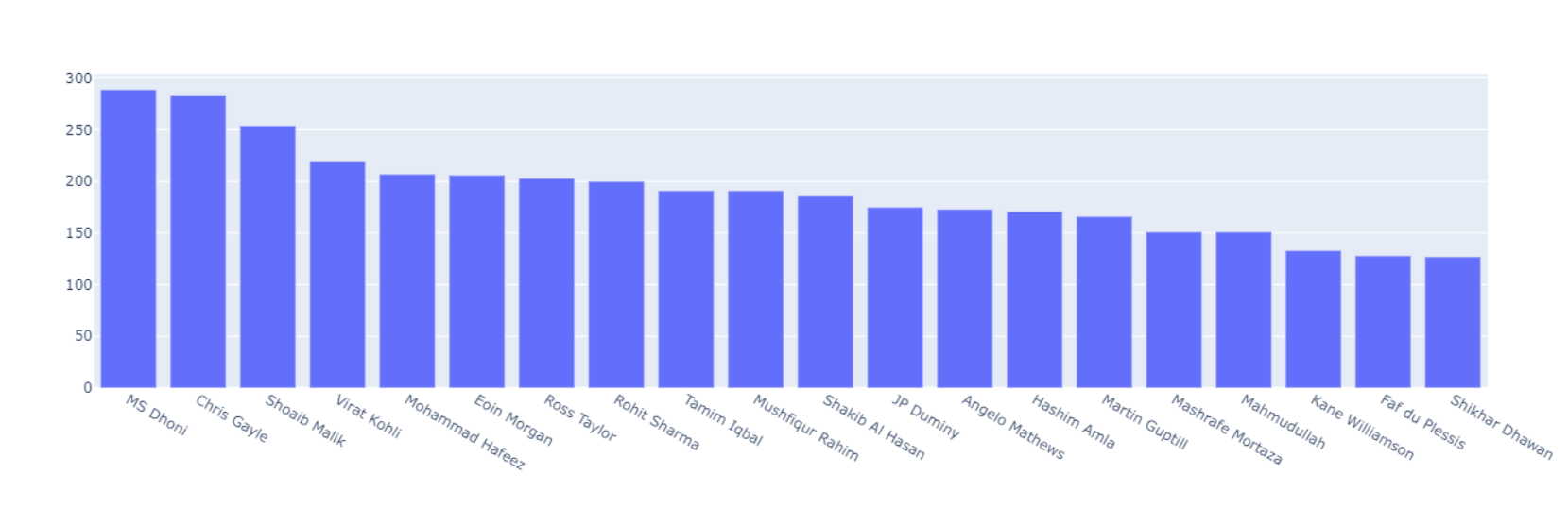


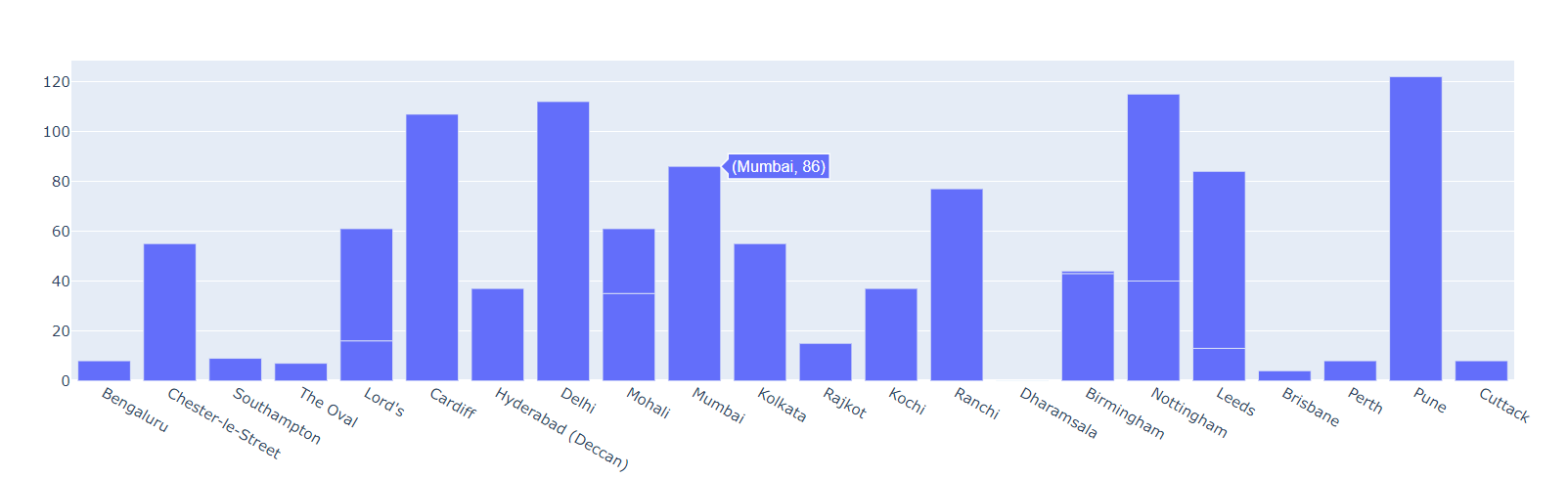
**F.1**

**F.2**



**F.3**

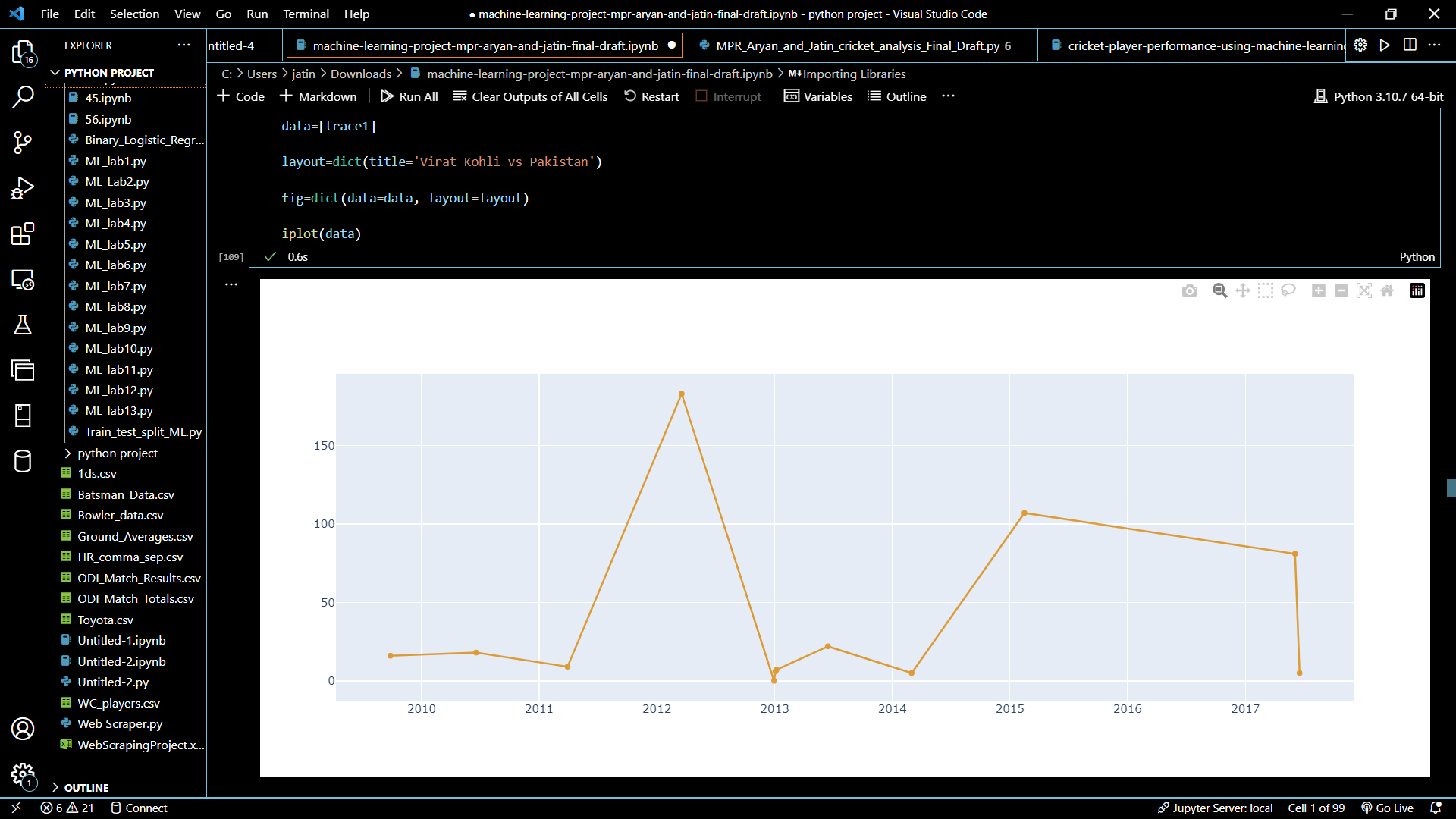
** F.4**

****

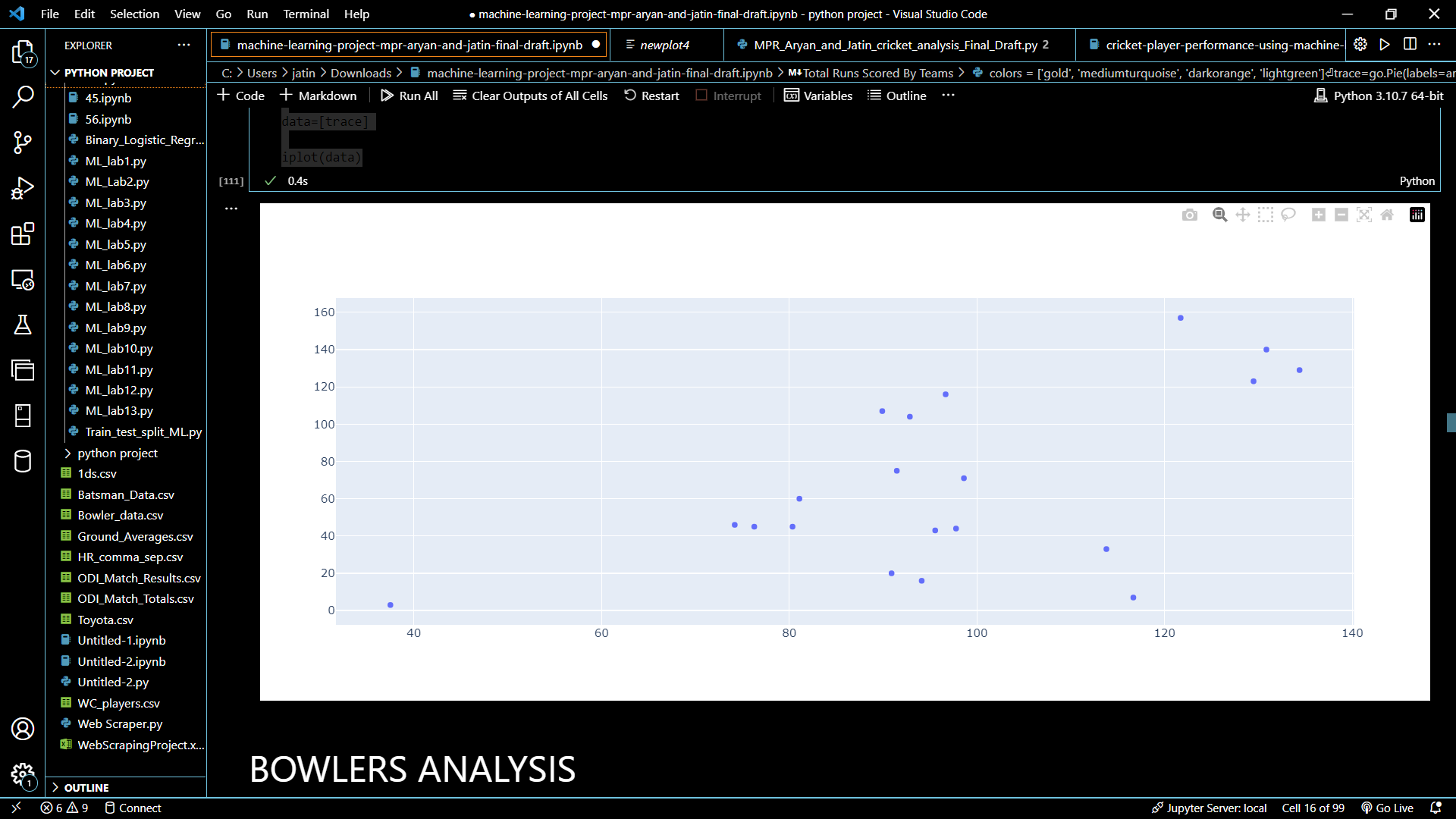
**F.5**



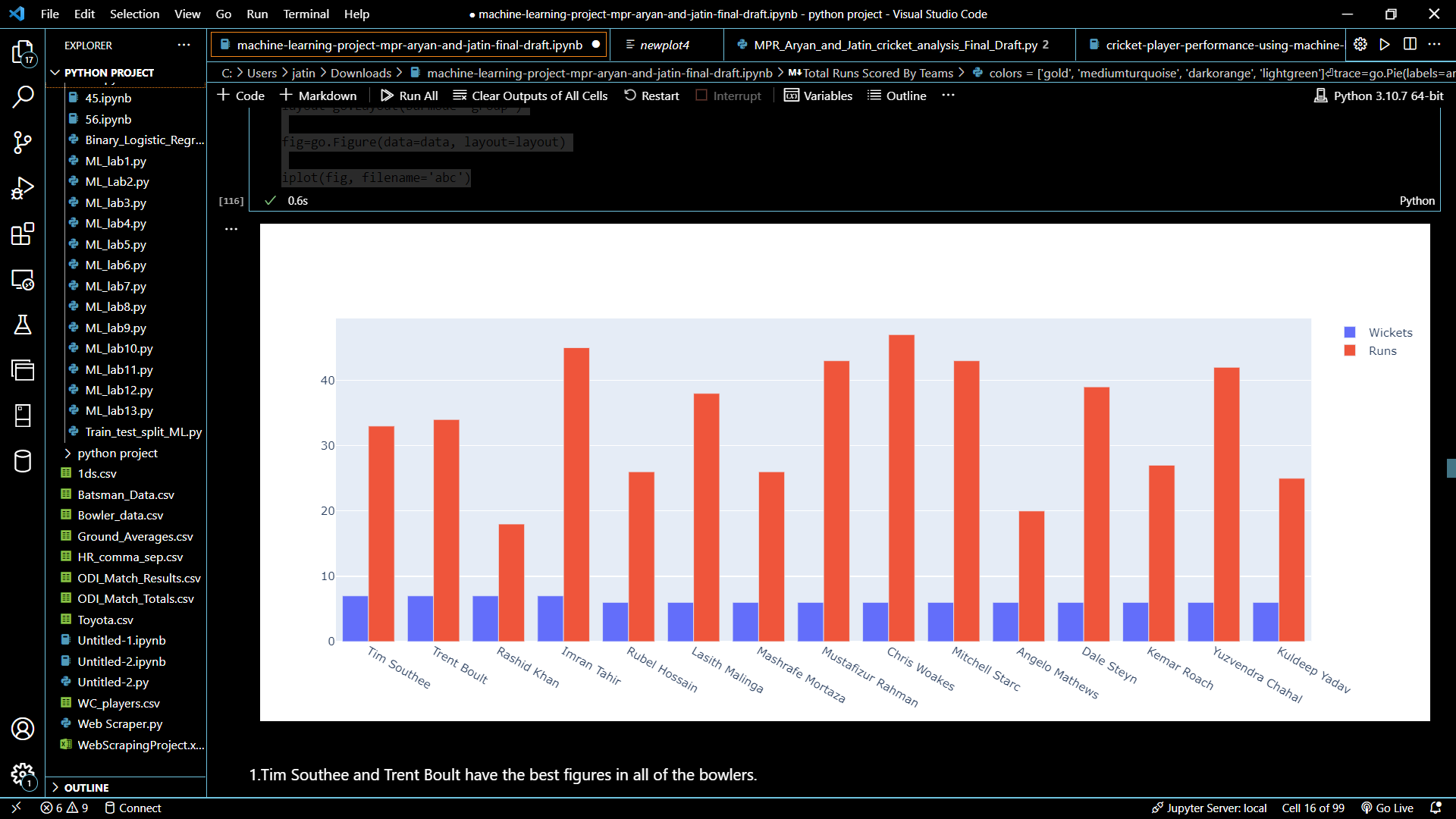
**F.6**



**F.7**

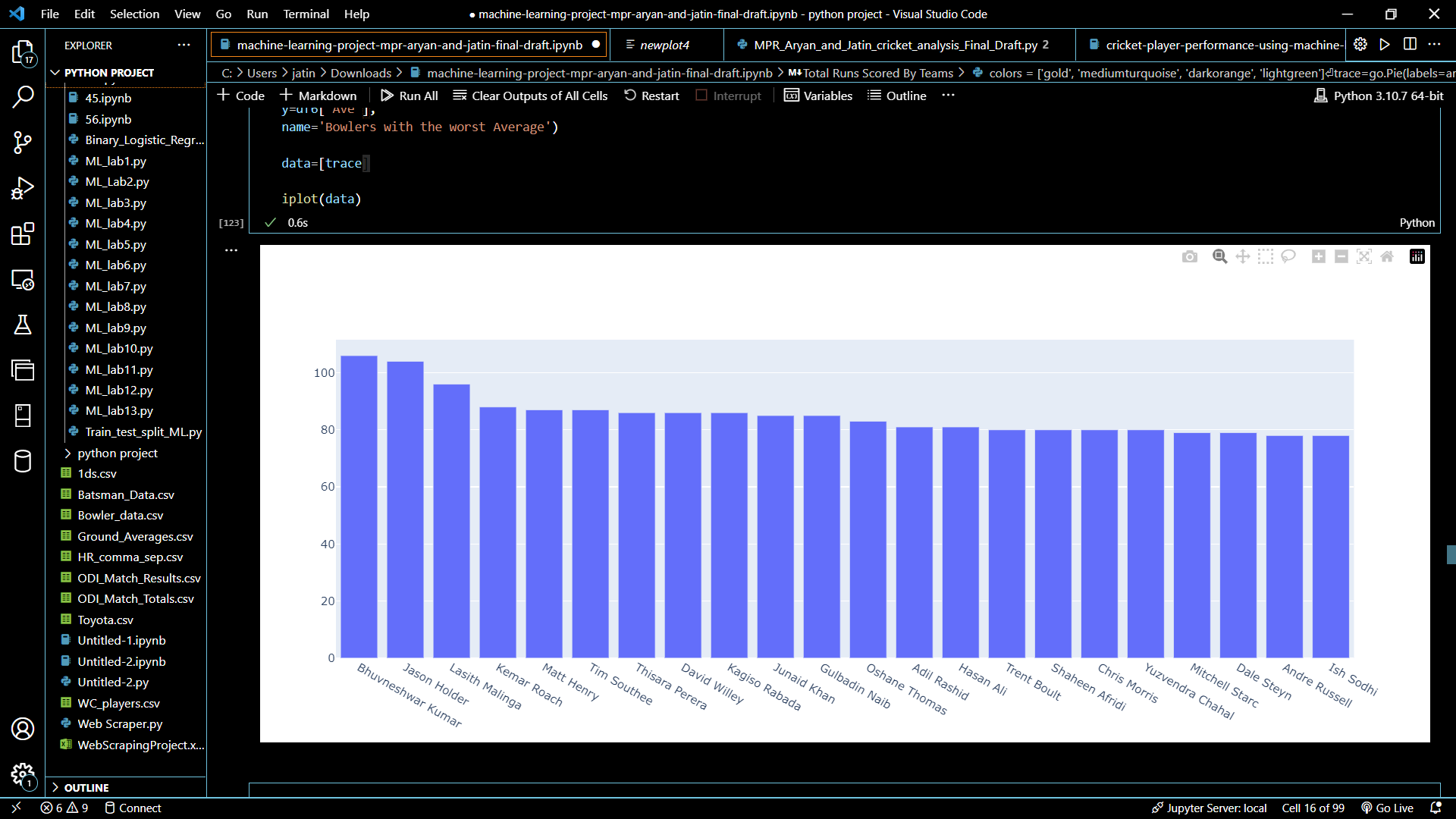


**F.8**



**F.9**

**F.10**



**F.11**

**F.12**

**CONCLUSION**

In the culmination of the "Cricket Statistics ML" project, our journey through advanced analytics and machine learning has unearthed valuable insights in cricket statistics. From the meticulous collection and preprocessing of data to the implementation of a robust linear regression model, we've bridged traditional statistics with cutting-edge predictive analytics.

Exploratory data analysis revealed intricate patterns, guiding our feature selection process and enhancing the model's predictive capabilities. The linear regression model not only forecasts player performances and match outcomes but also provides deeper insights into the game's underlying dynamics.

As we present our findings, we recognize this project as a milestone rather than a conclusion. It signifies the continuous evolution of cricket analytics, inviting further exploration and refinement. The "Cricket Statistics ML" project underscores our commitment to pushing the boundaries of sports analytics, contributing to the ever-growing body of knowledge in cricket statistics.

**FUTURE SCOPE**

1. **Integration of Advanced Models:**

* Explore and implement more sophisticated machine learning models, such as ensemble methods or deep learning, to capture intricate relationships within cricket data.

2. **Dynamic Player Performance Prediction:**

* Develop models that dynamically adapt to a player's form, fitness, and contextual factors, providing real-time predictions during tournaments or series.

3. **Team Dynamics Analysis:**

* Extend the project to analyze team dynamics, including the impact of player combinations, captaincy styles, and team strategies on match outcomes.

4. **Incorporation of Real-time Data:**

* Integrate real-time data feeds during live matches to enable instant updates and predictions, allowing for on-the-fly adjustments and insights.

5. **Player Injury Prediction:**

* Investigate the potential for predicting player injuries based on historical performance, workload, and fitness data, aiding teams in injury prevention strategies.

6. **Exploration of Unstructured Data:**

* Expand analysis to include unstructured data sources, such as player interviews, social media sentiments, and news articles, to understand the broader impact on player performance.

7. **Enhanced Visualization Techniques:**

* Develop advanced visualization techniques, including interactive dashboards and augmented reality interfaces, to make insights more accessible to a wider audience.

8. **Tournament Outcome Prediction:**

* Extend the model to predict overall tournament outcomes, considering factors like tournament format, venues, and historical team performances.

9. **Cross-Format Performance Analysis:**

* Adapt the model to analyze player performances across different formats of the game (Test, ODI, T20), considering the unique challenges and dynamics of each format.

10. **Collaboration with Cricket Experts:**

* Collaborate with cricket experts, coaches, and players to incorporate domain-specific knowledge, ensuring a more nuanced understanding of cricket dynamics.

11. **User Customization and Personalization:**

* Implement user customization features, allowing analysts to tailor the model to specific requirements and preferences, fostering a more personalized analytics experience.

12. **Predictive Analytics for Player Transfers:**

* Extend the model to predict the impact of player transfers on team dynamics, aiding teams in strategic decision-making during transfer windows.

13. **Global Performance Benchmarking:**

* Compare and benchmark player and team performances on a global scale, considering diverse cricketing conditions and oppositions.

14. **Continuous Model Refinement:**

* Establish a framework for continuous model refinement, incorporating feedback from cricket experts, analysts, and end-users to adapt to evolving trends and dynamics.

**BIBLIOGRAPHY**

**Books:**

1. The Art of Data Science" by Roger D. Peng and Elizabeth Matsui
2. Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili
3. Data Science from Scratch" by Joel Grus
4. Sports Analytics: A Guide for Coaches, Managers, and Other Decision Makers" by Benjamin C. Alamar

**Online Resources and Documentation:**

1. **Kaggle:**

Used for data science and machine learning related projects

1. **W3Schools:**

Interactive tutorials and references for data science and analytics technologies.

**Courses and Tutorials:**

* Machine Learning Tutorial Python | Codbasics Youtube channel
* Data Analytics Corporate Training Program | Jobaaj Learning
* Learn Plotly | Art of Vizualization

**Project-Specific Material:**

**Dataset:**

**Download Link:** <https://www.kaggle.com/datasets/saivamshi/cricket-world-cup-2019-players-data>