DATA SCIENCE MINOR PROJECT REPORT

DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING

PROJECT REPORT

(Project Semester January-April 2025)

Cricket Player Performance Analysis using Batting Data

Submitted by

Surendra Mahla

Registration No 12324358

Section KM004

Course Code INT-375

Under the Guidance of

GARGI SHARMA

UID: 29439

Discipline of CSE/IT

Lovely School of Computer Science

Lovely Professional University, Phagwara

DECLARATION

I, Surendra Mahla, student of Btech under CSE/IT Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 12/04/2025 Name: Surendra

Registration No. 12324358

TABLE OF CONTENT

Introduction

Cricket, being one of the most popular sports worldwide, has a massive following and generates vast volumes of data. In this project, we analyze a dataset that compiles batting statistics of top cricket players. The goal is to extract insights from performance metrics such as total runs, averages, strike rates, and centuries scored.

Exploratory Data Analysis (EDA) and Data Processing

To gain insights from the dataset, an in-depth Exploratory Data Analysis (EDA) was conducted using Python libraries such as Pandas, Matplotlib, and Seaborn. The major steps in the analysis included:

- Data Cleaning: Column names were stripped of leading/trailing whitespace. Columns were converted to numeric types where applicable.
- Missing Values: Filled using appropriate imputation (mean for numeric, mode for categorical).
- Feature Engineering: Created new columns like Runs per Innings, Strike to Average Ratio, and Total Centuries.
- Outlier Treatment: Noted for extreme scores but retained for analysis integrity.
- Data Visualization: Multiple graphs plotted for trends and player comparisons.

Analysis on Dataset

i. Introduction

We analyzed the dataset using Python libraries such as Pandas, Matplotlib, Seaborn, and Scikit-learn. The objective was to explore batting trends and build a simple prediction model for batting average using total runs.

ii. General Description

The dataset contains performance metrics of international cricket players. Primary fields include Player Name, Matches (Mat), Innings (Inns), Runs, High Score (HS), Average (Ave), Strike Rate (SR), Centuries (100), and Half-Centuries (50).

iii. Specific Requirements, Functions, and Formulas

- fillna(): Used to handle missing values.
- astype(), to_numeric(): For type conversions.
- groupby(), sort_values(): For sorting and subsetting.
- Formulas Used:
  - Runs per Innings = Runs / Innings
  - Strike to Avg = SR / Ave
  - Total Centuries = 100s + (0.5 * 50s)

iv. Analysis Results

- Top 10 run scorers identified.
- Most centuries: Player with maximum 100s.
- Positive correlation found between strike rate and average.
- Regression model showed linear relationship between Runs and Batting Average.

v. Visualization

1. Bar Plot of Top 10 Run Scorers
2. Bar Plot of Players with Most Centuries
3. Scatter Plot: Strike Rate vs Average
4. Bar Plot: Matches Played by Top Players
5. Bar Plot: Highest Individual Scores

5. Conclusion

The project successfully demonstrates how cricket performance metrics can be analyzed using data science tools. We identified key trends among top performers and developed a simple predictive model. The visualizations enhanced understanding of the underlying data structure.

6. Future Scope

- Add bowling and fielding metrics to make it multi-dimensional.
- Use time series data for performance trends over career span.
- Integrate clustering to group players by performance similarity.
- Apply ensemble models for better prediction accuracy.

7. References

[1] ESPN Cricinfo - https://stats.espncricinfo.com/

[2] Seaborn Documentation - https://seaborn.pydata.org/

[3] Scikit-learn Documentation - https://scikit-learn.org/

[4] Pandas Documentation - https://pandas.pydata.org/

Project Objectives

❑ To analyse batting performance data of international cricket players by leveraging data science tools and techniques for identifying top performers and trends.

❑ To preprocess and clean the dataset by handling missing values, converting data types, and performing feature engineering for effective analysis.

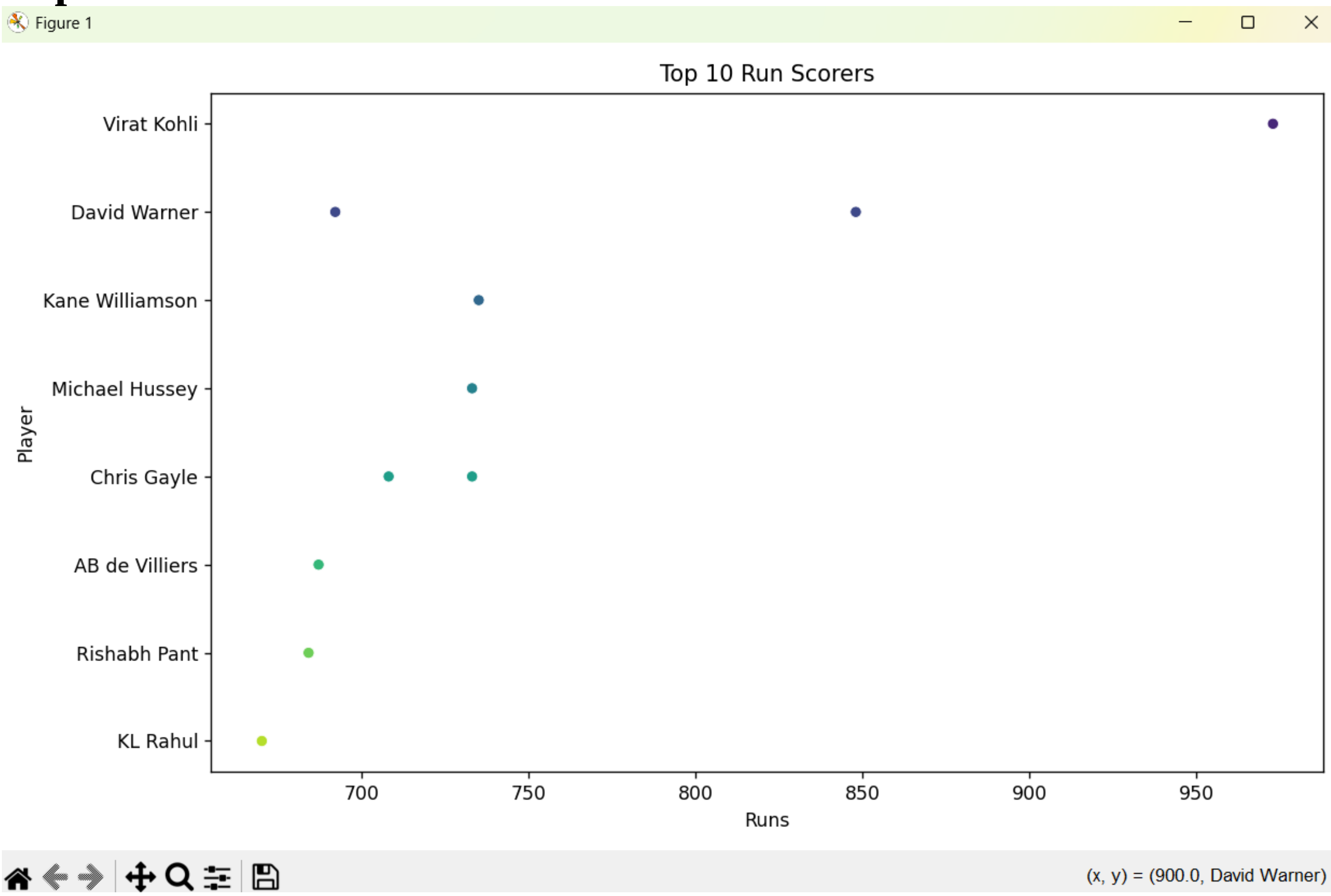
❑ To derive meaningful performance metrics, such as Runs per Innings and Strike-to-Average ratio, to better understand player efficiency.

❑ To visualize key performance indicators like total runs, centuries, and strike rates using bar charts, scatter plots, and other visual tools.

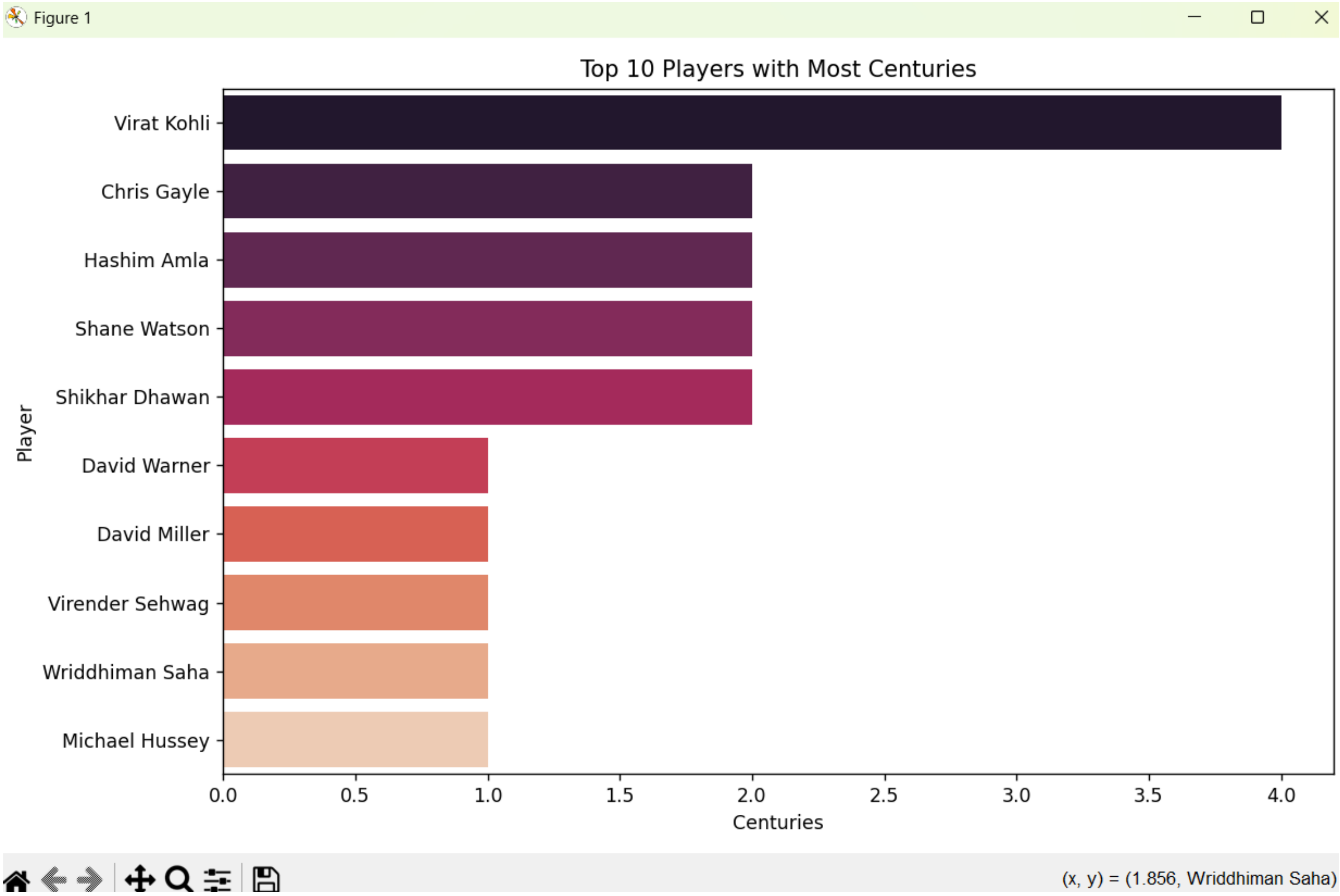
❑ To develop a simple regression model that predicts a player's batting average based on their total runs and other input metrics.

Graphs :

1. Top 10 Runs Scores



2. Top 10 Players with Most Centuries



3. Heat Map

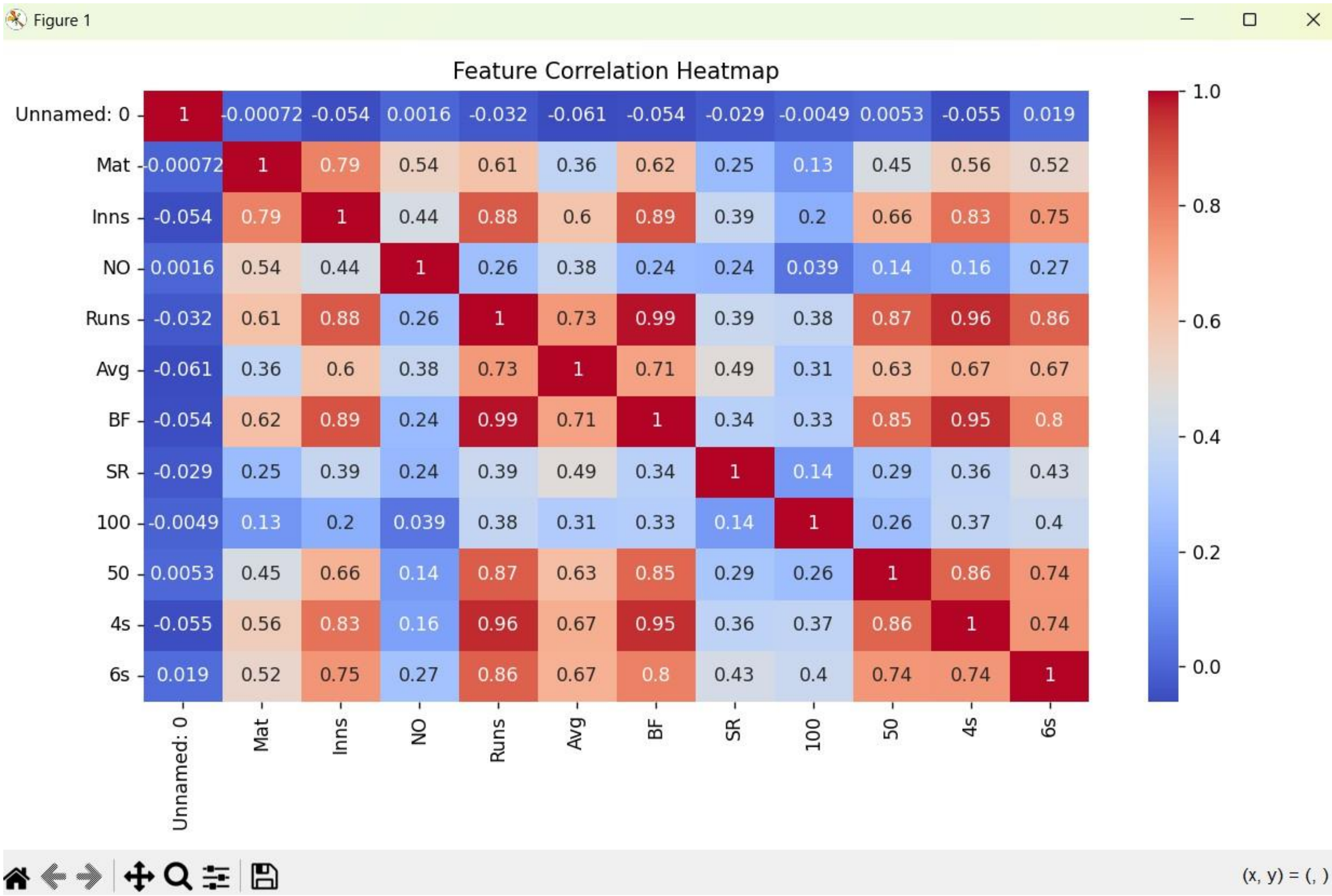
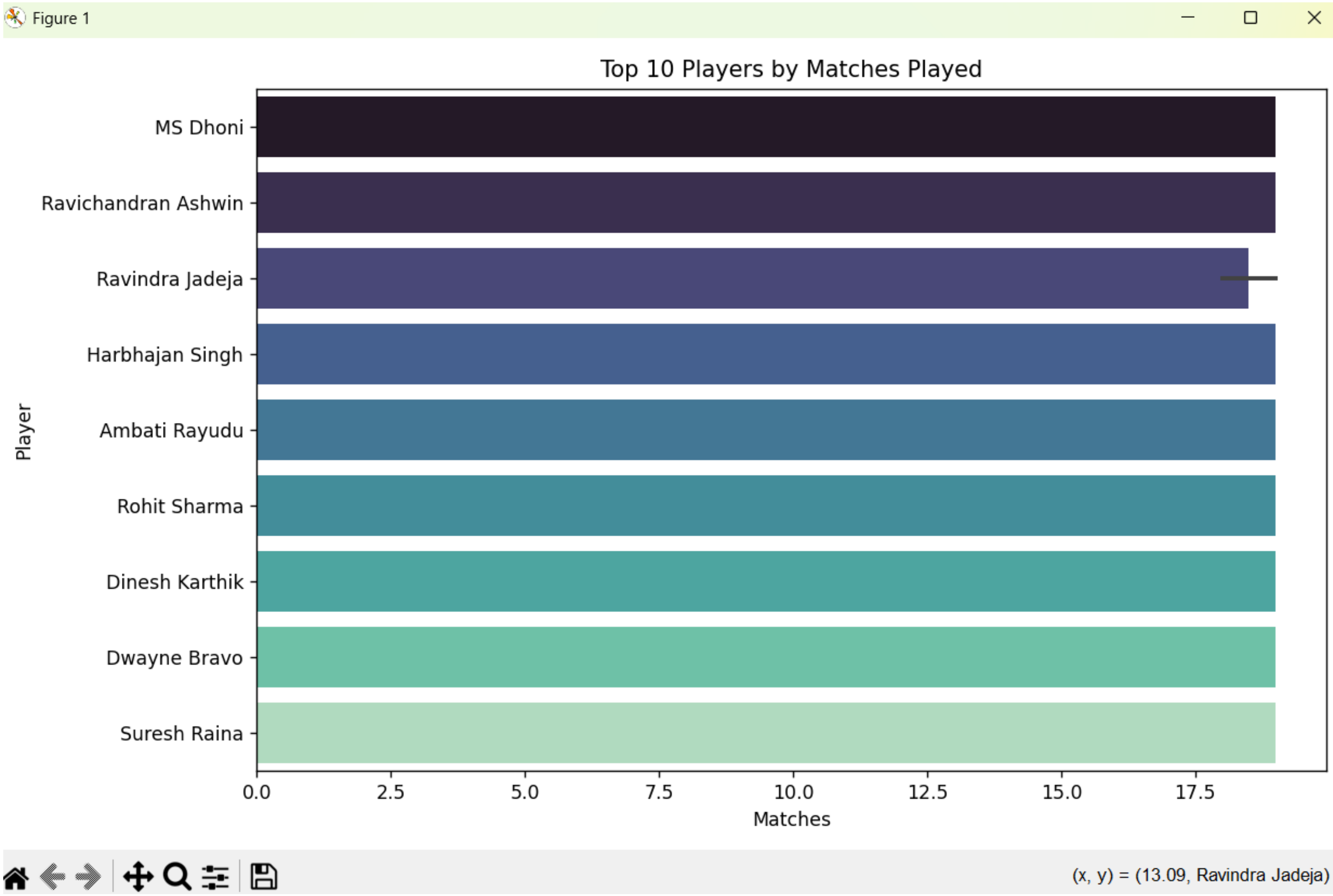
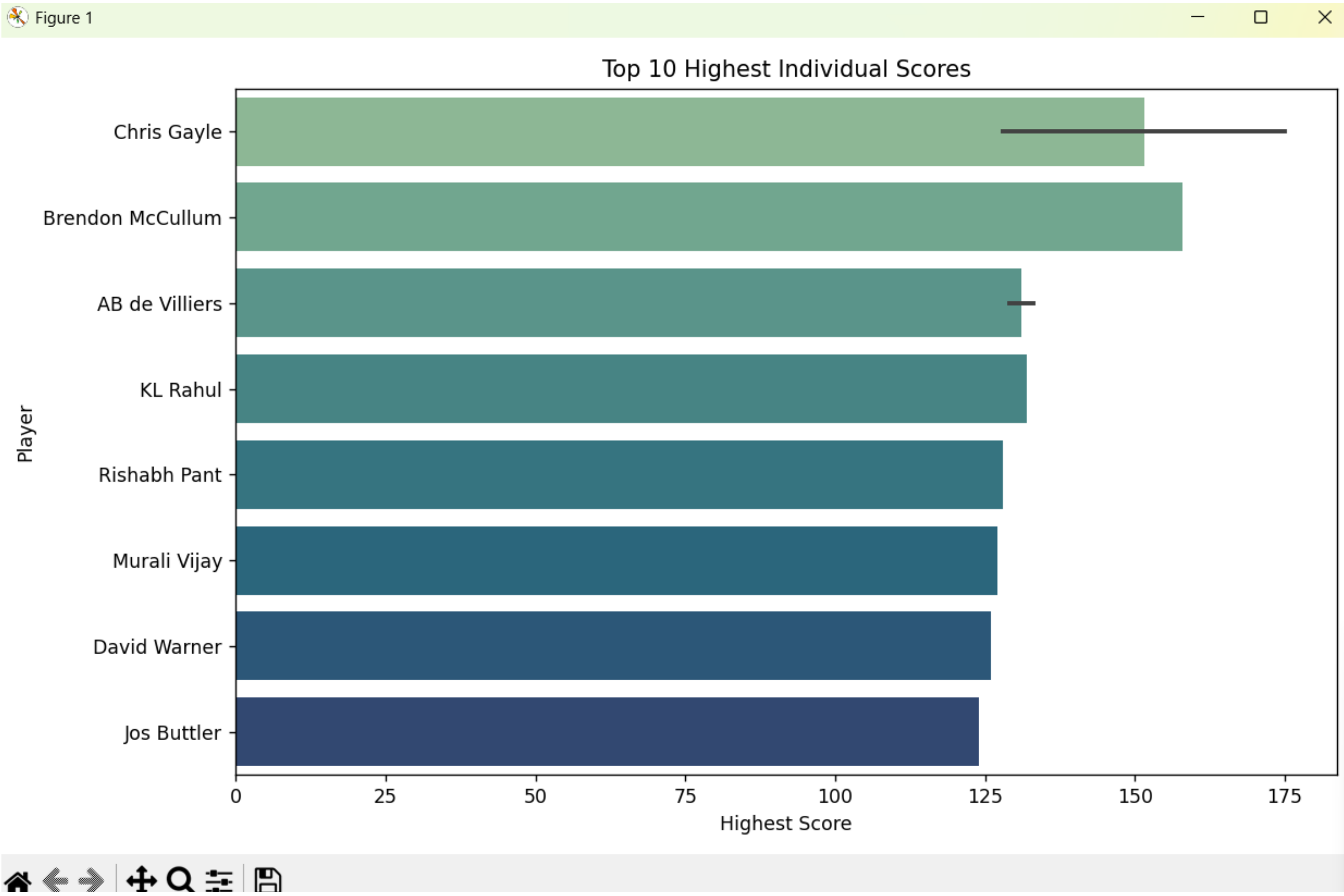


Figure 1
Feature Correlation Heatmap
Unnamed: 0 1 -0.00072 -0.054 0.0016 -0.032 -0.061 -0.054 -0.029 -0.0049 0.0053 -0.055 0.019
Mat -0.00072 1 0.79 0.54 0.61 0.36 0.62 0.25 0.13 0.45 0.56 0.52
Inns -0.054 0.79 1 0.44 0.88 0.6 0.89 0.39 0.2 0.66 0.83 0.75
NO 0.0016 0.54 0.44 1 0.26 0.38 0.24 0.24 0.039 0.14 0.16 0.27
Runs -0.032 0.61 0.88 0.26 1 0.73 0.99 0.39 0.38 0.87 0.96 0.86
Avg -0.061 0.36 0.6 0.38 0.73 1 0.71 0.49 0.31 0.63 0.67 0.67
BF -0.054 0.62 0.89 0.24 0.99 0.71 1 0.34 0.33 0.85 0.95 0.8
SR -0.029 0.25 0.39 0.24 0.39 0.49 0.34 1 0.14 0.29 0.36 0.43
100 -0.0049 0.13 0.2 0.039 0.38 0.31 0.33 0.14 1 0.26 0.37 0.4
50 0.0053 0.45 0.66 0.14 0.87 0.63 0.85 0.29 0.26 1 0.86 0.74
4s -0.055 0.56 0.83 0.16 0.96 0.67 0.95 0.36 0.37 0.86 1 0.74
6s 0.019 0.52 0.75 0.27 0.86 0.67 0.8 0.43 0.4 0.74 0.74 1
Unnamed: 0 Mat Inns NO Runs Avg BF SR 100 50 4s 6s
1.0 0.8 0.6 0.4 0.2 0.0
(x, y) = (,)

4.Top 10 Players by Matches Played



5. Top 10 Highest Individual Scores



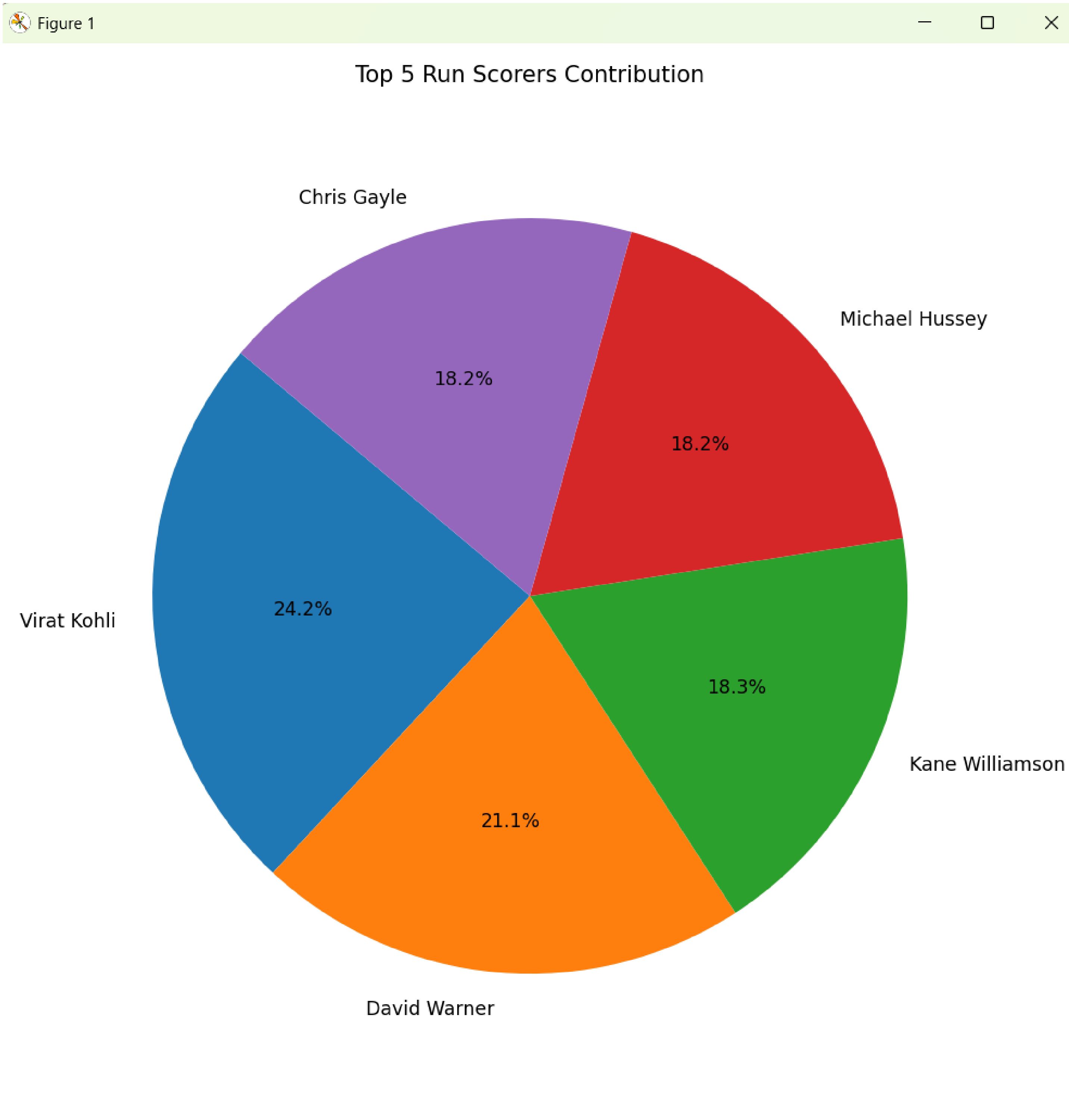


Figure 1
Top 5 Run Scorers Contribution
Chris Gayle
18.2%
Michael Hussey
18.2%
Virat Kohli
24.2%
18.3%
Kane Williamson
21.1%
David Warner

7. output

```
Columns in the dataset:
 Index(['Unnamed: 0', 'Player', 'Mat', 'Inns', 'NO', 'Runs', 'HS', 'Avg', 'BF',
       'SR', '100', '50', '4s', '6s'],
      dtype='object')

Sample data:
    Unnamed: 0             Player  Mat  Inns  NO  ...      SR 100  50  4s  6s
0           0        Shaun Marsh   11    11   2  ...  139.68   1   5  59  26
1           1     Gautam Gambhir   14    14   1  ...  140.89   0   5  68   8
2           2  Sanath Jayasuriya   14    14   2  ...  167.63   1   2  58  31
3           3       Shane Watson   15    15   5  ...  151.76   0   4  47  19
4           4       Graeme Smith   11    11   2  ...  121.82   0   3  54   8

[5 rows x 14 columns]

Missing Values After Cleaning:
 Unnamed: 0    0
Player        0
Mat           0
Inns          0
NO            0
Runs          0
HS            0
Avg           0
BF            0
SR            0
100           0
50            0
4s            0
6s            0
dtype: int64
Required columns ('Ave', 'Runs') not found in dataset
```

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

# Load dataset
data = pd.read_csv(r"C:\Users\Asus\Desktop\Python CA\Most Runs.csv")

# Strip column names to remove leading/trailing spaces
data.columns = data.columns.str.strip()

# Check the column names
print("Columns in the dataset:\n", data.columns)

# Show first few rows to visually inspect column values
print("\nSample data:\n", data.head())

# Fill missing values (basic strategy)
fill_columns = ['Player', 'Mat', 'Inns', 'Runs', 'Ave', 'SR', '100', '50']
for col in fill_columns:
    if col in data.columns:
        if data[col].dtype == 'O':
            data[col] = data[col].fillna(data[col].mode()[0])
        else:
            data[col] = data[col].fillna(data[col].mean())

# Check for any remaining nulls
print("\nMissing Values After Cleaning:\n", data.isnull().sum())

# Feature Engineering
if all(col in data.columns for col in ['Runs', 'Inns', 'SR', 'Ave', '100', '50']):
    data['Runs_per_Inns'] = data['Runs'] / data['Inns']
    data['Strike_to_Avg'] = data['SR'] / data['Ave']
    data['Total_Centuries'] = data['100'] + (data['50'] * 0.5)

# Graph 1: Top 10 Run Scorers

top10 = data.sort_values(by='Runs', ascending=False).head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x='Runs', y='Player', data=top10, hue='Player', palette='viridis', legend=False)
plt.title('Top 10 Run Scorers')
plt.xlabel('Runs')
plt.ylabel('Player')
plt.tight_layout()
plt.show()
```

```python
# ------------------------------------------
# 2. Players with Most Centuries
top100 = data.sort_values(by='100', ascending=False).head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x='100', y='Player', hue='Player', data=top100, palette='rocket', legend=False)
plt.title('Top 10 Players with Most Centuries')
plt.xlabel('Centuries')
plt.ylabel('Player')
plt.tight_layout()
plt.show()


# Graph 3: Correlation Heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(data.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title('Feature Correlation Heatmap')
plt.tight_layout()
plt.show()

# Graph 4: Histogram of Averages
if 'Ave' in data.columns:
    plt.figure(figsize=(10, 5))
    sns.histplot(data['Ave'], bins=30, kde=True)
    plt.title('Distribution of Batting Averages')
    plt.xlabel('Average')
    plt.tight_layout()
    plt.show()



# 4. Most Matches Played
top_matches = data.sort_values(by='Mat', ascending=False).head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x='Mat', y='Player', hue='Player', data=top_matches, palette='mako', legend=False)
plt.title('Top 10 Players by Matches Played')
plt.xlabel('Matches')
plt.ylabel('Player')
plt.tight_layout()
plt.show()

# ------------------------------------------
# 5. Highest Individual Scores
data['HS_numeric'] = data['HS'].astype(str).str.replace('*', '', regex=False)
data['HS_numeric'] = pd.to_numeric(data['HS_numeric'], errors='coerce')
top_hs = data.sort_values(by='HS_numeric', ascending=False).head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x='HS_numeric', y='Player', hue='Player', data=top_hs, palette='crest', legend=False)
plt.title('Top 10 Highest Individual Scores')
plt.xlabel('Highest Score')
plt.ylabel('Player')
plt.tight_layout()
plt.show()
```

```python
    # Normalize
    scaler_X = MinMaxScaler()
    scaler_y = MinMaxScaler()
    X_scaled = scaler_X.fit_transform(X)
    y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1))

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.1, random_state=42)

    # Train linear regression
    model = LinearRegression()
    model.fit(X_train, y_train)

    print(f"\nModel Coefficients:\nSlope: {model.coef_[0][0]:.2f}, Intercept: {model.intercept_[0]:.2f}")

    # Evaluate
    y_pred_train = model.predict(X_train)
    y_pred_test = model.predict(X_test)
    print(f"Train MSE: {mean_squared_error(y_train, y_pred_train):.4f}")
    print(f"Test MSE: {mean_squared_error(y_test, y_pred_test):.4f}")

    # Plot regression
    plt.figure(figsize=(10, 6))
    plt.scatter(X_scaled, y_scaled, alpha=0.5, label='Data')
    plt.plot(X_scaled, model.predict(X_scaled), color='red', linewidth=2, label='Regression Line')
    plt.title('Regression: Runs vs Average')
    plt.xlabel('Normalized Runs')
    plt.ylabel('Normalized Average')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    # Predict function
    def predict_average(runs):
        runs_scaled = scaler_X.transform([[runs]])
        avg_scaled = model.predict(runs_scaled)
        avg_original = scaler_y.inverse_transform(avg_scaled)[0][0]
        return round(avg_original, 2)

    # Example prediction
    example_runs = 10500
    print(f"\nPredicted Average for {example_runs} runs: {predict_average(example_runs)}")

    # Optional: User input
    try:
        user_input = float(input("Enter runs to predict average: "))
        print(f"Predicted Batting Average: {predict_average(user_input)}")
    except ValueError:
        print("Invalid input. Please enter a numeric value.")
else:
    print("Required columns ('Ave', 'Runs') not found in dataset.")
```