```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("deliveries.csv")
df_copy=df
```

```
df.head()
df.columns
```

```
    Index(['match_id', 'season', 'start_date', 'venue', 'innings', 'ball',
           'batting_team', 'bowling_team', 'striker', 'non_striker', 'bowler',
           'runs_off_bat', 'extras', 'wides', 'noballs', 'byes', 'legbyes',
           'penalty', 'wicket_type', 'player_dismissed', 'other_wicket_type',
           'other_player_dismissed'],
          dtype='object')
```

```
columns_to_keep = ['match_id', 'venue','batting_team', 'bowling_team','striker',
df_selected = df[columns_to_keep]
```

```
selected_entries = df_selected[df['bowler'] == 'TA Boult']

df_bowler = pd.DataFrame(selected_entries)

df_bowler.reset_index(drop=True, inplace=True)
df_bowler
```

	match_id	venue	batting_team	bowling_team	striker	bowler	runs_
0	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult	
1	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult	
2	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult	
3	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	DJ Malan	TA Boult	
4	1	Narendra Modi Stadium,	England	New Zealand	JM Bairstow	TA Boult	

```
match_id_column = 'match_id'
batting_team_column = 'batting_team'
bowling_team_column = 'bowling_team'
striker_column = 'striker'
bowler_column = 'bowler'
runs_off_bat_column = 'runs_off_bat'
extras_column = 'extras'

unique_bowlers = df[bowler_column].unique()
bowler_data = {}
for bowler in unique_bowlers:
    bowler_data[bowler] = df[df[bowler_column] == bowler][[match_id_column, batt

# Access the data for a specific bowler (replace 'BowlerName' with the actual bo
specific_bowler_data = bowler_data.get('TA Boult', pd.DataFrame())

# If you want to reset the index of each bowler's DataFrame
```

```
for bowler, data in bowler_data.items():
    data.reset_index(drop=True, inplace=True)

# Display the data for a specific bowler
print(specific_bowler_data)
```

```
     match_id  batting_team bowling_team      striker    bowler  runs_off_b
0           1       England  New Zealand  JM Bairstow  TA Boult
1           1       England  New Zealand  JM Bairstow  TA Boult
2           1       England  New Zealand  JM Bairstow  TA Boult
3           1       England  New Zealand     DJ Malan  TA Boult
4           1       England  New Zealand  JM Bairstow  TA Boult
..        ...           ...          ...          ...       ...            .
397        32  South Africa  New Zealand    DA Miller  TA Boult
398        32  South Africa  New Zealand    DA Miller  TA Boult
399        32  South Africa  New Zealand    H Klaasen  TA Boult
400        32  South Africa  New Zealand    DA Miller  TA Boult
401        32  South Africa  New Zealand    H Klaasen  TA Boult

     extras
0         0
1         0
2         0
3         0
4         0
..      ...
397       0
398       0
399       0
400       0
401       0

[402 rows x 7 columns]
```

```
#Scraping batsman data
from bs4 import BeautifulSoup
import requests
url = 'https://www.espncricinfo.com/records/tournament/bowling-best-career-econo
response = requests.get(url)

soup = BeautifulSoup(response.text, "html.parser")

x = soup.find_all('table')[0]
# print(len(x))
y = x.find_all('tr')
df_economy = []
for i in y:
  temp = []
  for j in i.find_all('td'):
    #  print(j.text,end=" ")
    temp.append(j.text)
  df_economy.append(temp)
df_economy=pd.DataFrame(df_economy)

def extract_first_two_words(text):
    words = text.split()
    return ' '.join(words[:2])
df_economy.columns = df_economy.iloc[0]
df_economy = df_economy.reindex(df_economy.index.drop(0))

# # Apply the function to the specified column
df_economy['Player'] = df_economy['Player'].apply(lambda x: extract_first_two_wo
merged_df = pd.merge(df_selected, df_economy, left_on='bowler', right_on='Player
merged_df = merged_df.drop(columns = ['Player','Span'])

merged_df = merged_df.rename(columns={'Ave':'Ave_bowl' , 'Runs':'Runs_given','SR
merged_df.columns
```

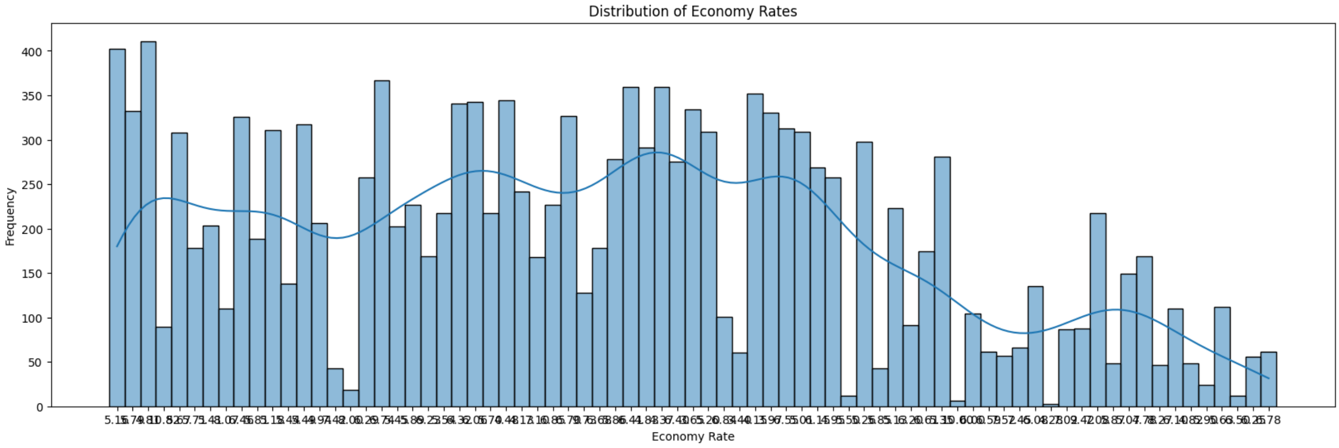
```
Index(['match_id', 'venue', 'batting_team', 'bowling_team', 'striker',
       'bowler', 'runs_off_bat', 'extras', 'wicket_type', 'Mat_bowl',
'Overs',
       'Mdns', 'Balls', 'Runs_given', 'Wkts', 'BBI', 'Ave_bowl', 'Econ',
       'SR_bowl', '4', '5', '10'],
      dtype='object')
```

EDA for Bowler-Centric Analysis:

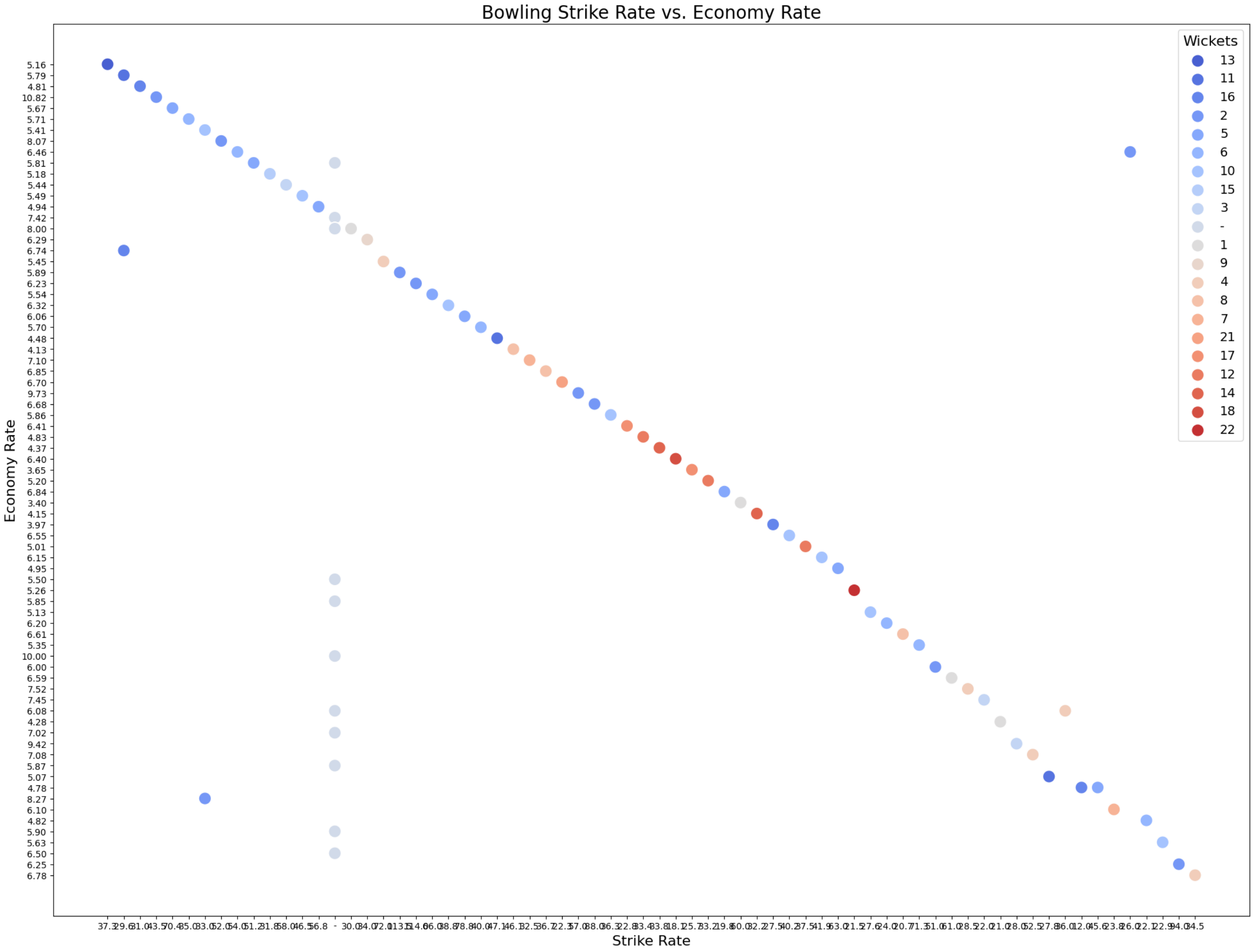
```
print(merged_df.describe())
print(merged_df.info())
```

```
           match_id  runs_off_bat        extras  wicket_type
count  14513.000000  14513.000000  14513.000000        411.0
mean      16.425550      0.894577      0.044305          1.0
std        9.201236      1.397221      0.281322          0.0
min        1.000000      0.000000      0.000000          1.0
25%        8.000000      0.000000      0.000000          1.0
50%       17.000000      0.000000      0.000000          1.0
75%       25.000000      1.000000      0.000000          1.0
max       32.000000      6.000000      5.000000          1.0
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14513 entries, 0 to 14512
Data columns (total 22 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   match_id      14513 non-null  int64  
 1   venue         14513 non-null  object 
 2   batting_team  14513 non-null  object 
 3   bowling_team  14513 non-null  object 
 4   striker       14513 non-null  object 
 5   bowler        14513 non-null  object 
 6   runs_off_bat  14513 non-null  int64  
 7   extras        14513 non-null  int64  
 8   wicket_type   411 non-null    float64
 9   Mat_bowl      14513 non-null  object 
 10  Overs         14513 non-null  object 
 11  Mdns          14513 non-null  object 
 12  Balls         14513 non-null  object 
 13  Runs_given    14513 non-null  object 
 14  Wkts          14513 non-null  object 
 15  BBI           14513 non-null  object 
 16  Ave_bowl      14513 non-null  object 
 17  Econ          14513 non-null  object 
 18  SR_bowl       14513 non-null  object 
 19  4             14513 non-null  object 
 20  5             14513 non-null  object 
 21  10            14513 non-null  object 
dtypes: float64(1), int64(3), object(18)
memory usage: 2.5+ MB
None
```

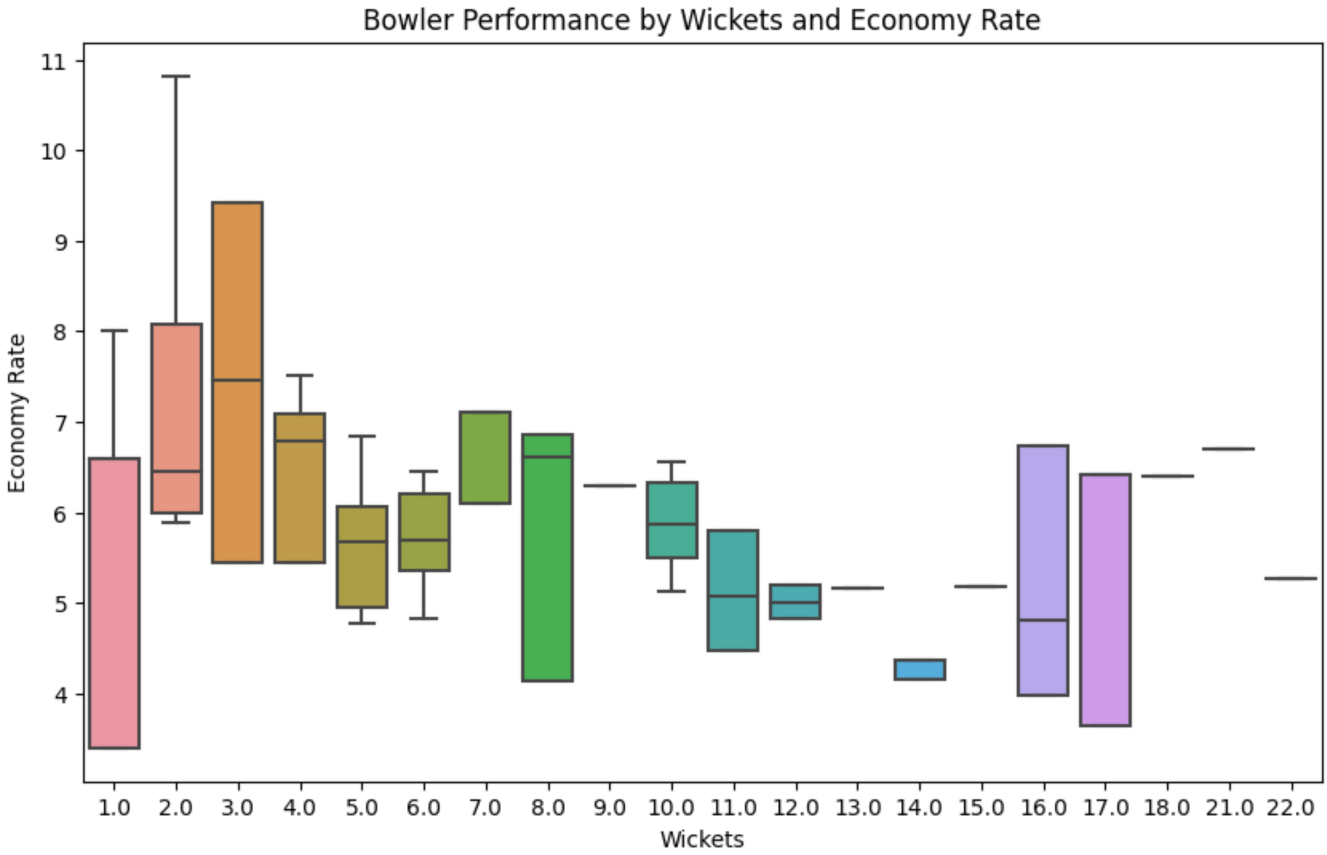
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(28, 6))
sns.histplot(merged_df['Econ'], bins=20, kde=True)
plt.title('Distribution of Economy Rates')
plt.xlabel('Economy Rate')
plt.ylabel('Frequency')
plt.show()
```



```
plt.figure(figsize=(24, 18))
scatter = sns.scatterplot(x='SR_bowl', y='Econ', data=merged_df, hue='Wkts', pal

plt.title('Bowling Strike Rate vs. Economy Rate', fontsize=20)
plt.xlabel('Strike Rate', fontsize=16)
plt.ylabel('Economy Rate', fontsize=16)

# Resize the legend
plt.legend(title='Wickets', fontsize=14, title_fontsize=16, markerscale=2) # Adj
plt.show()
```


Bowling Strike Rate vs. Economy Rate
Economy Rate
5.16
5.79
4.81
10.82
5.67
5.71
5.41
8.07
6.46
5.81
5.18
5.44
5.49
4.94
7.42
8.00
6.29
6.74
5.45
5.89
6.23
5.54
6.32
6.06
5.70
4.48
4.13
7.10
6.85
6.70
9.73
6.68
5.86
6.41
4.83
4.37
6.40
3.65
5.20
6.84
3.40
4.15
3.97
6.55
5.01
6.15
4.95
5.50
5.26
5.85
5.13
6.20
6.61
5.35
10.00
6.00
6.59
7.52
7.45
6.08
4.28
7.02
9.42
7.08
5.87
5.07
4.78
8.27
6.10
4.82
5.90
5.63
6.50
6.25
6.78
Strike Rate
37.3
29.6
31.0
43.5
70.4
35.0
33.0
52.0
54.0
51.2
31.8
58.0
46.5
56.8
-
30.0
34.0
72.0
113.5
14.0
66.0
38.8
78.8
40.0
47.1
46.1
32.5
36.7
22.3
57.0
88.0
36.3
22.8
33.4
33.8
18.1
25.7
33.2
19.8
60.0
32.2
27.5
40.2
37.5
41.9
63.0
21.5
27.6
24.0
20.7
71.3
51.0
61.0
28.5
22.0
21.0
28.0
52.5
27.8
36.0
12.0
45.6
23.8
26.0
22.1
22.9
94.0
34.5
Wickets
● 13
● 11
● 16
● 2
● 5
● 6
● 10
● 15
● 3
● -
● 1
● 9
● 4
● 8
● 7
● 21
● 17
● 12
● 14
● 18
● 22

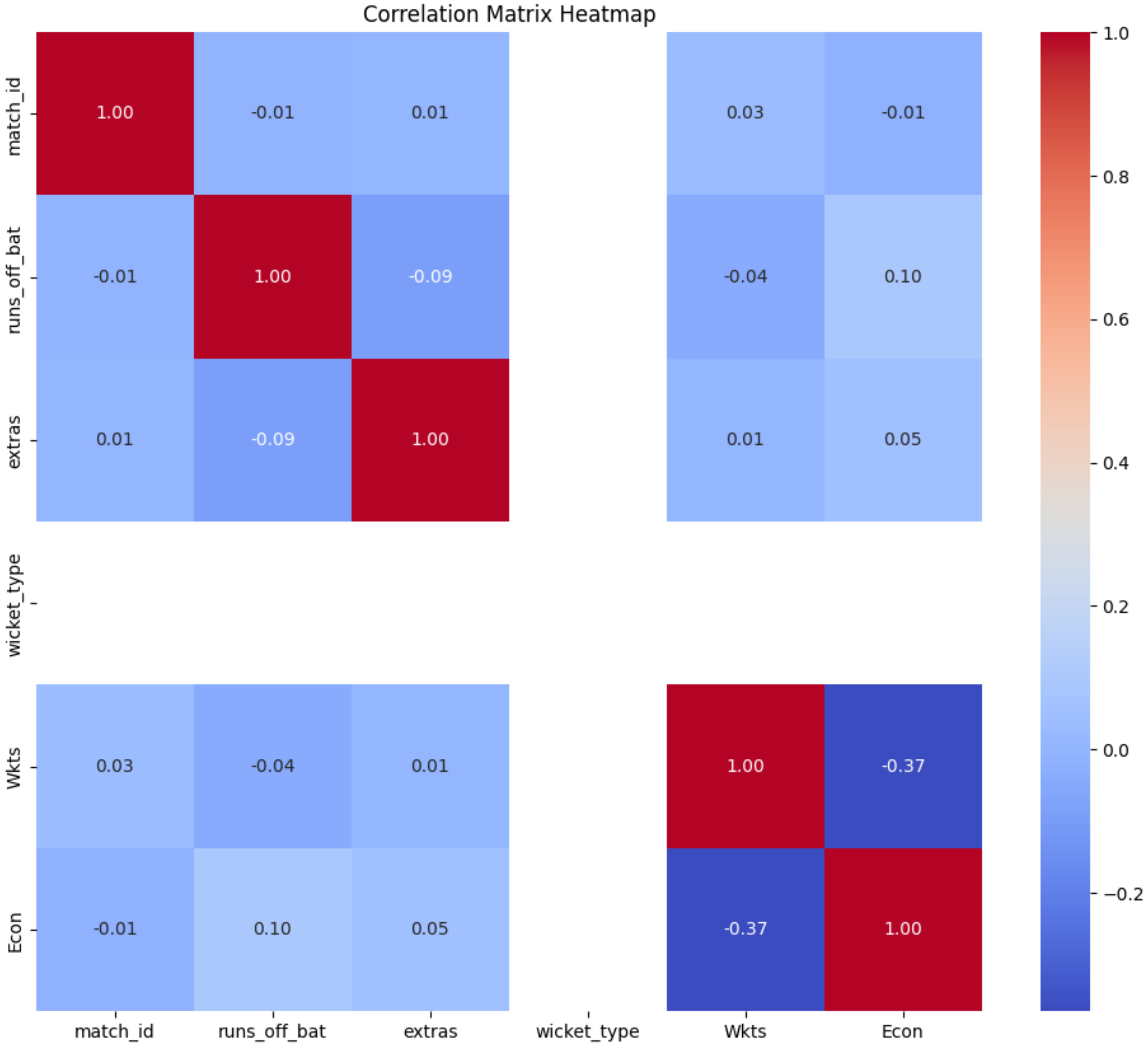
```
# Convert 'Wkts' and 'Econ' columns to numeric (if not already numeric)
merged_df['Wkts'] = pd.to_numeric(merged_df['Wkts'], errors='coerce')
merged_df['Econ'] = pd.to_numeric(merged_df['Econ'], errors='coerce')

# Drop rows with NaN values in 'Wkts' or 'Econ' (if any)
merged_df = merged_df.dropna(subset=['Wkts', 'Econ'])

plt.figure(figsize=(10, 6))
sns.boxplot(x='Wkts', y='Econ', data=merged_df)
plt.title('Bowler Performance by Wickets and Economy Rate')
plt.xlabel('Wickets')
plt.ylabel('Economy Rate')
plt.show()
```



```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
correlation_matrix = merged_df.corr()

plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix Heatmap')
plt.show()
```

```
<ipython-input-82-a83ce528690f>:4: FutureWarning: The default value of nume
  correlation_matrix = merged_df.corr()
```



```
url = 'https://www.espncricinfo.com/records/tournament/batting-highest-career-ba

response = requests.get(url)

soup = BeautifulSoup(response.text, "html.parser")

x = soup.find_all('table')[0]
# print(len(x))
y = x.find_all('tr')
df_batsman = []
for i in y:
  temp = []
  for j in i.find_all('td'):
    #  print(j.text,end=" ")
    temp.append(j.text)
  df_batsman.append(temp)
df_batsman=pd.DataFrame(df_batsman)

def extract_first_two_words(text):
    words = text.split()
    return ' '.join(words[:2])
df_batsman.columns = df_batsman.iloc[0]
df_batsman = df_batsman.reindex(df_batsman.index.drop(0))
# # # Apply the function to the specified column
df_batsman['Player'] = df_batsman['Player'].apply(lambda x: extract_first_two_wo
merged_df = pd.merge(merged_df, df_batsman, left_on='striker', right_on='Player'
merged_df = merged_df.drop(columns = ['Player','Span'])
merged_df
# df_batsman
```

	match_id	venue	batting_team	bowling_team	striker	bowler	ru
0	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult	
1	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult	
2	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult	
3	1	Narendra Modi Stadium,	England	New Zealand	JM Bairstow	TA Boult	

		Ahmedabad				
4	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult
...	...	...	...	...	...	...
10906	5	MA Chidambaram Stadium, Chepauk, Chennai	Australia	India	C Green	RA Jadeja
10907	5	MA Chidambaram Stadium, Chepauk, Chennai	Australia	India	C Green	RA Jadeja
10908	5	MA Chidambaram Stadium, Chepauk, Chennai	Australia	India	C Green	RA Jadeja
10909	5	MA Chidambaram Stadium, Chepauk, Chennai	Australia	India	C Green	RA Jadeja
10910	5	MA Chidambaram Stadium, Chepauk, Chennai	Australia	India	C Green	RA Jadeja

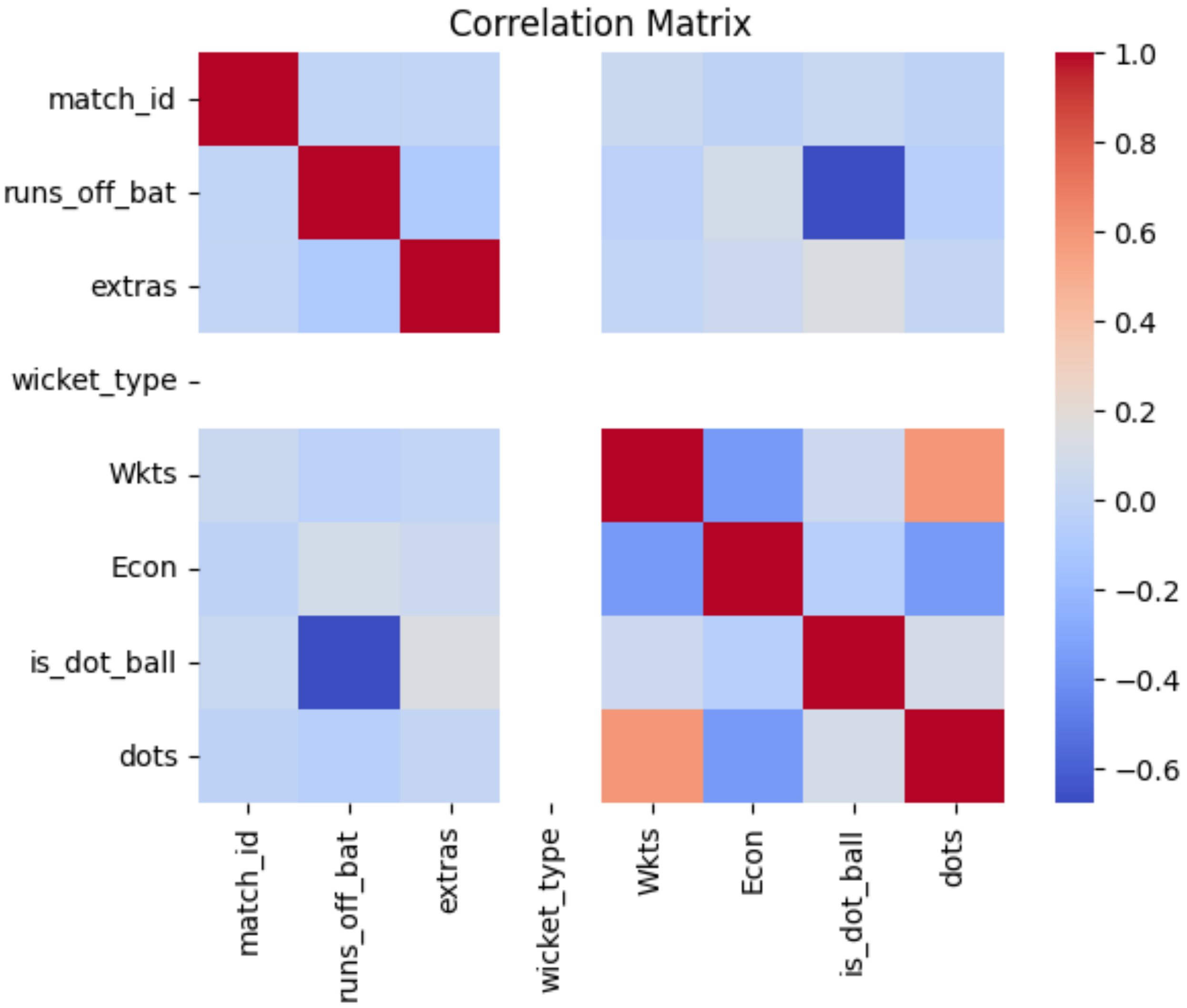
10911 rows × 35 columns

```
merged_df.to_csv('output.csv', index=False)
```

```
# # Assuming your dataset is stored in a DataFrame named 'df'
# # You can load your dataset using pd.read_csv or another appropriate method
team1 = 'New Zealand'
team2 = 'India'
# Extract data for the specific match between IND and AUS
# a_vs_b_match = merged_df[(merged_df['batting_team'].isin([team1, team2])) & (m

# Create a binary column 'is_dot_ball' indicating whether the run_off_bat is 0 (
merged_df['is_dot_ball'] = (merged_df['runs_off_bat'] == 0)
dot_balls_count = merged_df.groupby('bowler')['is_dot_ball'].sum().reset_index()
dot_balls_count =  dot_balls_count.rename(columns = {'bowler':'temp', 'is_dot_ba
dot_balls_count
merged_df = pd.merge(merged_df, dot_balls_count, left_on='bowler', right_on='tem
merged_df = merged_df.drop(columns='temp')
```

```
corr_matrix = merged_df.corr()
sns.heatmap(corr_matrix,annot=False, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
<ipython-input-86-c6d5ddf2f0c8>:1: FutureWarning: The default value of nume
  corr_matrix = merged_df.corr()
```



```
merged_df
```

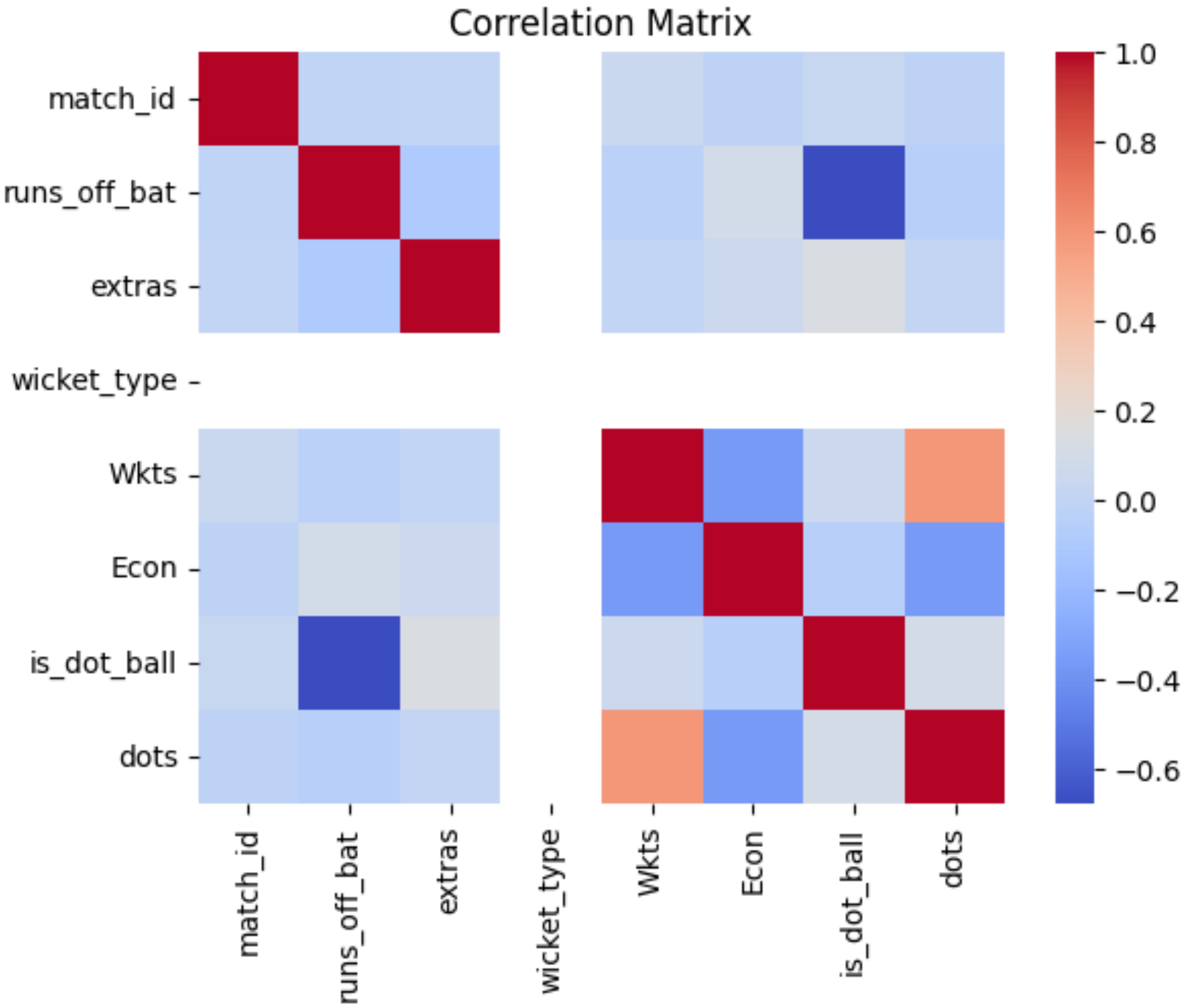
	match_id	venue	batting_team	bowling_team	striker	bowler	ru
0	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult	
1	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult	

		Ahmedabad				
2	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult
3	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult
4	1	Narendra Modi Stadium, Ahmedabad	England	New Zealand	JM Bairstow	TA Boult
...	...	...	...	...	...	...
10906	2	Rajiv Gandhi International Stadium, Uppal, Hyd...	Pakistan	Netherlands	Mohammad Rizwan	Saqib Zulfiqar
10907	2	Rajiv Gandhi International Stadium, Uppal, Hyd...	Pakistan	Netherlands	Mohammad Rizwan	Saqib Zulfiqar
10908	2	Rajiv Gandhi International Stadium, Uppal, Hyd...	Pakistan	Netherlands	Mohammad Rizwan	Saqib Zulfiqar
10909	2	Rajiv Gandhi International Stadium, Uppal, Hyd...	Pakistan	Netherlands	Mohammad Rizwan	Saqib Zulfiqar
10910	2	Rajiv Gandhi International Stadium, Uppal, Hyd...	Pakistan	Netherlands	Mohammad Rizwan	Saqib Zulfiqar

10911 rows × 37 columns

```
corr_matrix = merged_df.corr()
sns.heatmap(corr_matrix,annot=False, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
    <ipython-input-88-c6d5ddf2f0c8>:1: FutureWarning: The default value of nume
      corr_matrix = merged_df.corr()
```



```
dot_balls_count = merged_df.groupby('bowler')['is_dot_ball'].sum().reset_index()

dot_balls_count = dot_balls_count.rename(columns = { 'is_dot_ball' : 'dots_count
# final_df = pd.merge(dot_balls_count,merged_df,left_on='bowler2',right_on='bowl
final_df = pd.merge(dot_balls_count,merged_df,on='bowler')
dot_balls_count
```

	bowler	dots_count
0	A Dutt	143
1	A Zampa	79
2	AAP Atkinson	25
3	AD Mathews	25
4	AU Rashid	128
...	...	...
64	TA Boult	167
65	TG Southee	1
66	Taskin Ahmed	114
67	Usama Mir	67
68	V Kohli	0

69 rows × 2 columns

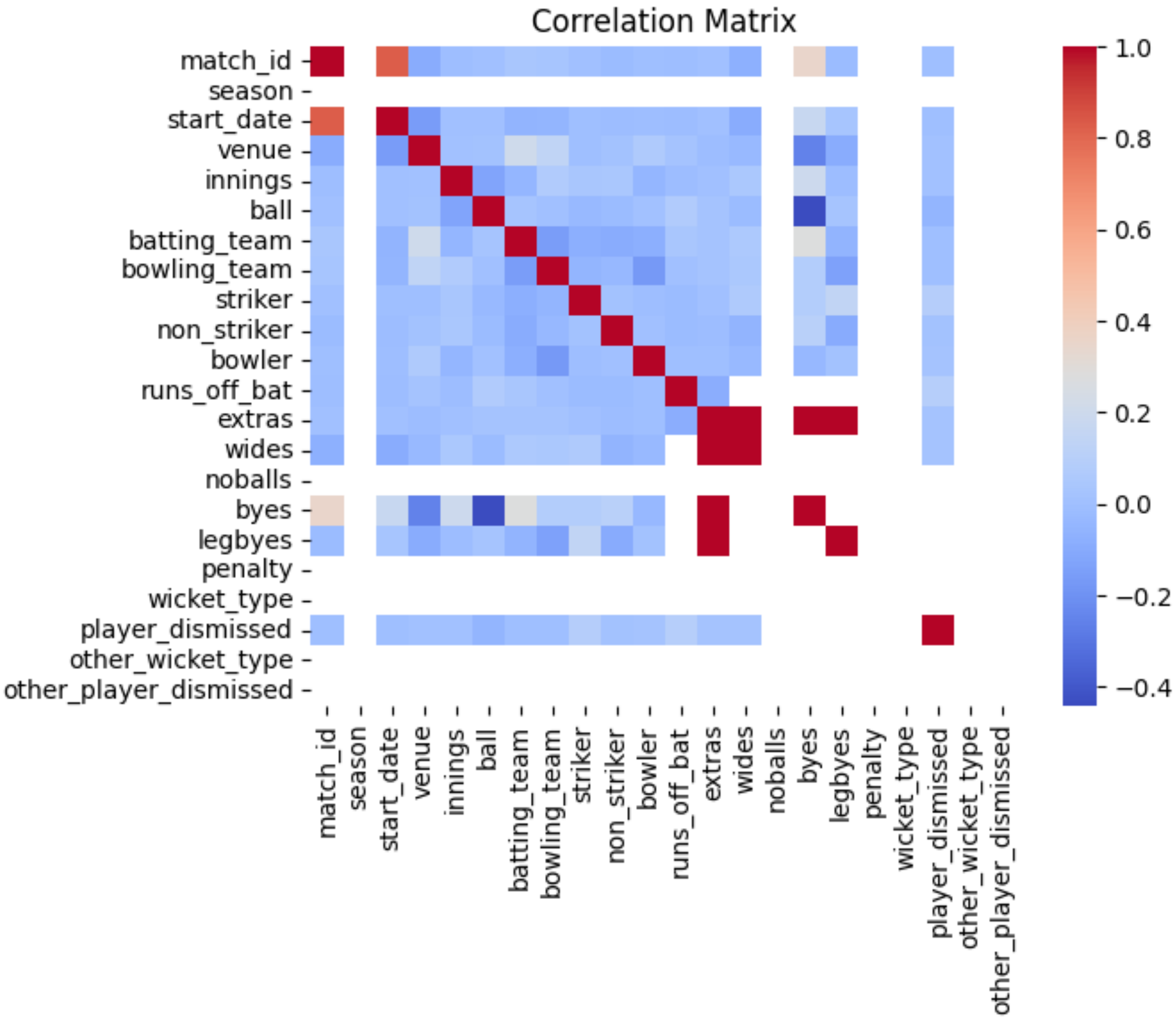
```
df_economy
```

	Player	Span	Mat	Overs	Mdns	Balls	Runs	Wkts	BBI	Ave	Econ	S
1	R Ashwin	2023-2023	1	10.0	1	60	34	1	1/34	34.00	3.40	60
2	JJ Bumrah	2023-2023	9	72.5	6	437	266	17	4/39	15.64	3.65	25
3	RA Jadeja	2023-2023	9	73.3	4	441	292	16	5/33	18.25	3.97	27
4	Mohammad Nabi	2023-2023	9	61.3	4	369	254	8	3/28	31.75	4.13	46
5	Kuldeep Yadav	2023-2023	9	75.1	2	451	312	14	2/7	22.28	4.15	32
...	...	...	...	...	...	...	...	...	...	...	...	
96	Agha Salman	2023-2023	3	5.0	-	30	46	-	-	-	9.20	
97	Hasan Mahmud	2023-2023	2	14.0	-	84	132	3	2/67	44.00	9.42	28
	M	2023-										

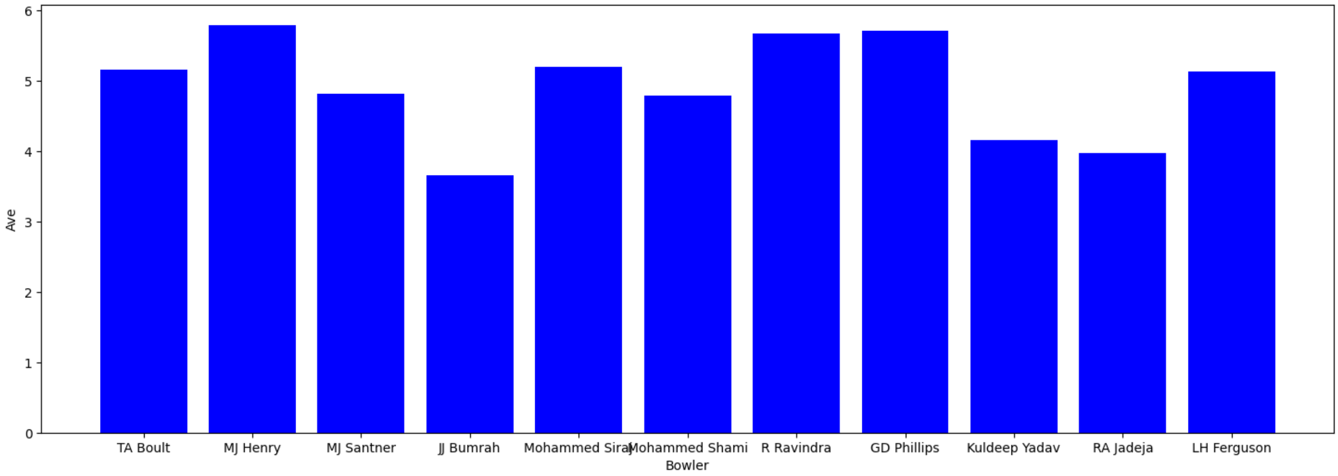
```
final_df = pd.merge(dot_balls_count,df_economy,left_on='bowler',right_on='Player
final_df = final_df.astype(int, errors='ignore')
final_df = final_df.replace('-',0)
```

```
from sklearn.preprocessing import LabelEncoder

# Assuming 'df' is your DataFrame
le = LabelEncoder()
df_transformed = df
for column in df_transformed.columns:
    if df_transformed[column].dtype == 'object':
        df_transformed[column] = le.fit_transform(df_transformed[column])
```

```
corr_matrix = df_transformed.corr()
sns.heatmap(corr_matrix, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



```
df_economy
team1 = 'New Zealand'
team2 = 'India'
# Extract data for the specific match between IND and AUS
a_vs_b_match = merged_df[(merged_df['batting_team'].isin([team1, team2])) & (me
df_unique = a_vs_b_match.drop_duplicates(subset=['bowler'])
df_unique
plt.figure(figsize=(18, 6))
plt.bar(df_unique['bowler'], df_unique['Econ'], color='blue')

# Add labels and title
plt.xlabel('Bowler')
plt.ylabel('Ave')
# Show the plot
plt.show()
```



```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_absolute_error,r2_score


features = ['Runs','Wkts','Econ','Ave','SR','Mdns','dots_count','Balls']
data = final_df[features]


X_train, X_test, y_train, y_test = train_test_split(data.drop('dots_count', axis

# Train a RandomForestRegressor (since it's a regression problem)
regressor = RandomForestRegressor()
regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = regressor.predict(X_test)

# Evaluate the model using Mean Absolute Error (MAE) instead of accuracy
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")
r_squared = r2_score(y_test, y_pred)
print(f"R2 score: {r_squared}")
import pickle
pickle_rfc = open("most_dots_bowler.pkl","wb")
pickle.dump(regressor, pickle_rfc)
pickle_rfc.close()

regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = regressor.predict(X_test)

# Evaluate the model using Mean Absolute Error (MAE) instead of accuracy
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")
r_squared = r2_score(y_test, y_pred)
print(f"R2 score: {r_squared}")
```

```
Mean Absolute Error: 13.590714285714284
R2 score: 0.8783775789184485
Mean Absolute Error: 20.701212772403682
R2 score: 0.7185300978894799
```

Hyper-parameter Tuning

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create the RandomForestRegressor
regressor = RandomForestRegressor(random_state=42)

# Create RandomizedSearchCV object
random_search = RandomizedSearchCV(estimator=regressor, param_distributions=par
                                   n_iter=100, scoring='neg_mean_absolute_error
                                   cv=5, verbose=1, random_state=42, n_jobs=-1)

# Fit the RandomizedSearchCV to the data
random_search.fit(X_train, y_train)

# Get the best parameters and the best model
best_params = random_search.best_params_
best_score = random_search.best_score_
best_regressor = random_search.best_estimator_

# Make predictions on the test set
y_pred = best_regressor.predict(X_test)

# Evaluate the model using Mean Absolute Error (MAE) and R2 score
mae = mean_absolute_error(y_test, y_pred)
r_squared = r2_score(y_test, y_pred)

# Save the best RandomForestRegressor model using pickle
with open("best_rfc_model.pkl", "wb") as pickle_rfc:
    pickle.dump(best_regressor, pickle_rfc)

print(f"Best Parameters: {best_params}")
print(f"Best Mean Absolute Error: {-best_score}")

print(f"RandomForestRegressor - Mean Absolute Error: {mae}")
print(f"RandomForestRegressor - R2 score: {r_squared}")
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Best Parameters: {'n_estimators': 100, 'min_samples_split': 10, 'min_sample
Best Mean Absolute Error: 16.656791753938283
RandomForestRegressor - Mean Absolute Error: 13.329699109554257
RandomForestRegressor - R2 score: 0.8622498971729458
```