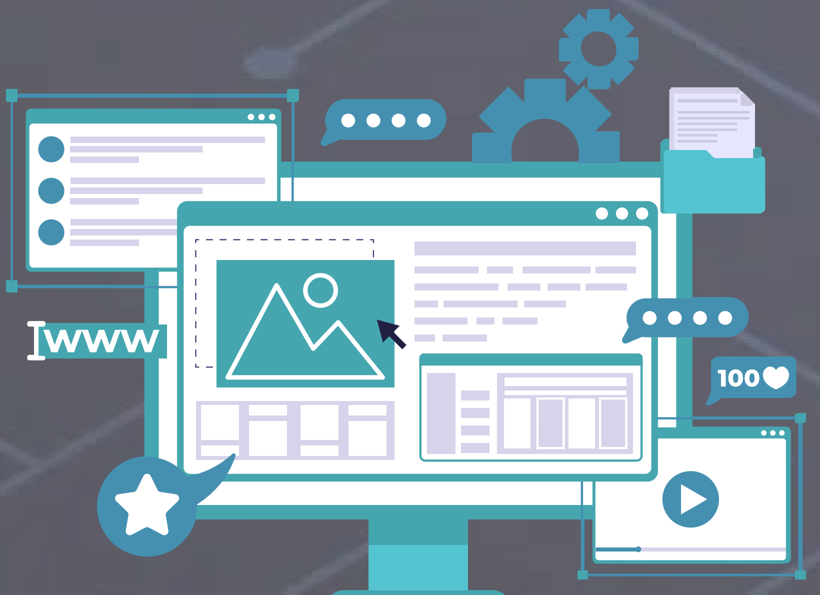
MIT | Academy of Engineering

EDS Project on :

ICC Tect Batting Figures



GROUP: JAYANTIKA KARNA 229
PRATHMESH LASHKARE 237
AKASH JARAD 228

Introducton

- Data analytics is a field of study and practice that focuses on extracting meaningful information and patterns ,trends and coorelation from raw data to support decision-making.
- In the context of the Batting Figures dataset, data analysis becomes a window through which we can do various analysis such as data manipulation, data visualization ,etc.
- By applying data analytics techniques, we aim get inforamtion of stats of cricket player

Motivation

As we know that, the cricket is most popular sport in world.the dataset of ICC test batting figures provide information of cricket player on their best score ,not outs,Centuries,Half-centuries.

Details of Dataset

Name:- ICC Test Batting Figures
Number of Features :- 11
Numbers of Records:-3002

Data Manipulation

Data manipulation is a fundamental process in data analysis that involves transforming and preparing raw data to make it suitable for further exploration and analysis. It encompasses a range of operations aimed at ensuring data quality, consistency, and usability.

```
#Q2.Find Player who had did maximum centuries in test matches?
print("Find Player who had did maximum centuries in test matches:-",data['Player']
[data['Centuries'].max()],"\n","Number Of  Century:-",data['Centuries'].max(),"\n\n")

#Q3.Find Player who had did maximum Half-centuries in test matches?
print("Find Player who had did maximum Half-centuries in test matches:-",data['Player']
[data['Fifties'].max()],"\n","Number of Half-Centuru:-",data['Fifties'].max(),"\n\n")

#Q4.Find median of all players Batting Average?
print("Find median of all players Batting Average:-",data['Batting Average'].median(),"\n\n")

#Q5.Find mean of all players Batting Average?
print("Find median of all players Batting Average:-",data['Batting Average'].mean(),"\n\n")
```

Data Manipulation

```
#Q6.show player name who had debut before 1900 and also count this player?
d1=data['Player'][data['Debut Year']<=1900]
print("show player name who had debut before 1900:-\n",d1,"\n")
print("Count:-",len(d1),"\n\n")

#Q7. show player name who had played upto 1900 and also count this player?
d2=data['Player'][data['Upto year played']<=1900]
print("show player name who had played upto 1900:-\n",d2,"\n")
print("Count:-",len(d2),"\n\n")

#Q8. show player name who had played from india and also count this player?
d3=data['Player'][data['Country']=='India']
print("show player name who had played from india:-\n",d3,"\n")
print("Count:-",len(d3),"\n\n")
```

Data Visualization

Data visualization is the process of representing data and information visually through charts, graphs, maps, and other graphical elements.Data visualization transforms complex data into visual representations that enhance understanding, reveal patterns, and support decision-making.

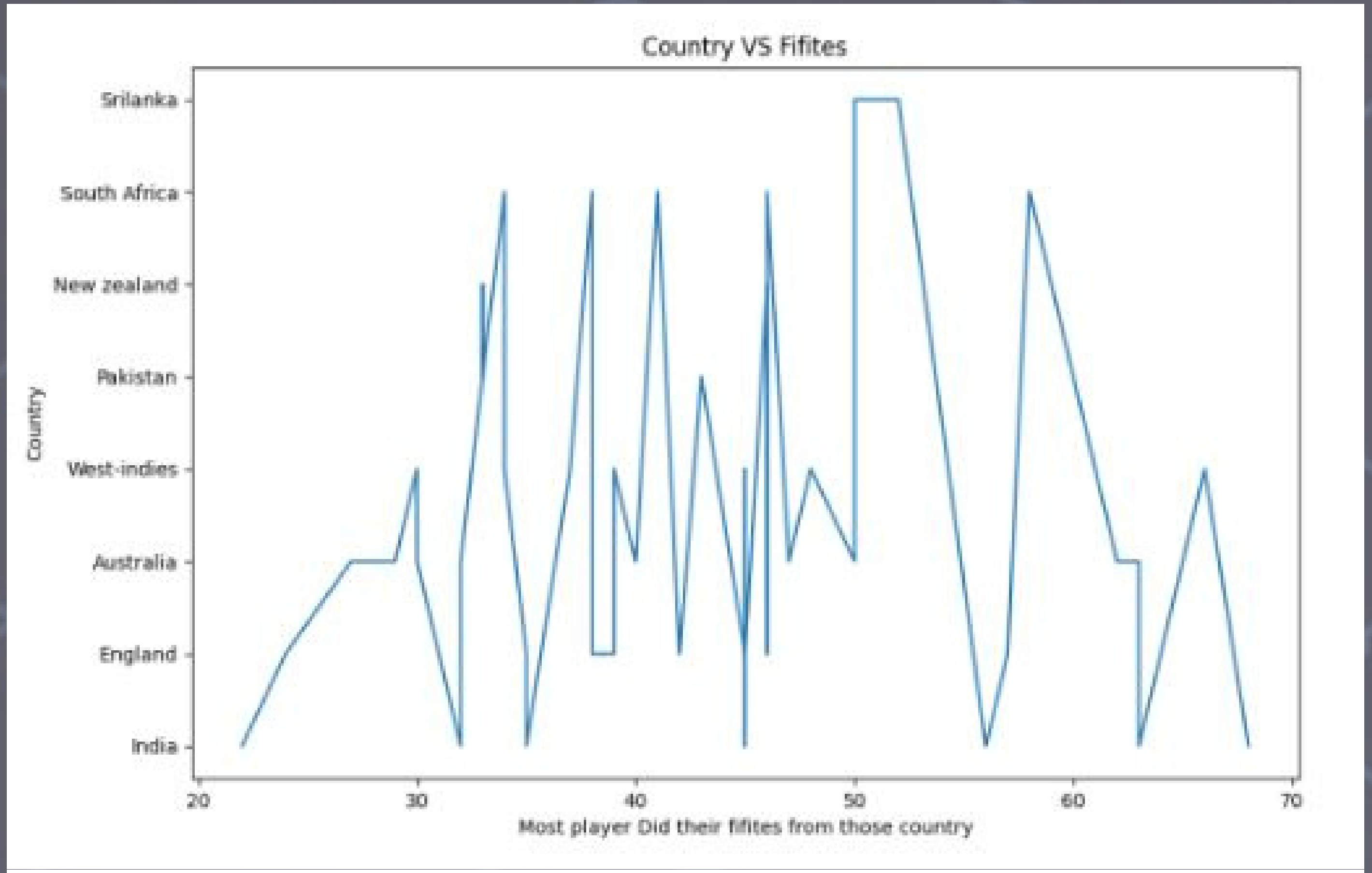
```
#Line chart of Country VS Matches Played
data=data.sort_values(by='Matches Played')
mat_x=data['Matches Played']
con_y=data['Country']
plt.plot(mat_x,con_y)
plt.title('Country VS Matches Played')
plt.xlabel('Most player played matches from those countries')
plt.ylabel('Country')
plt.show()
```

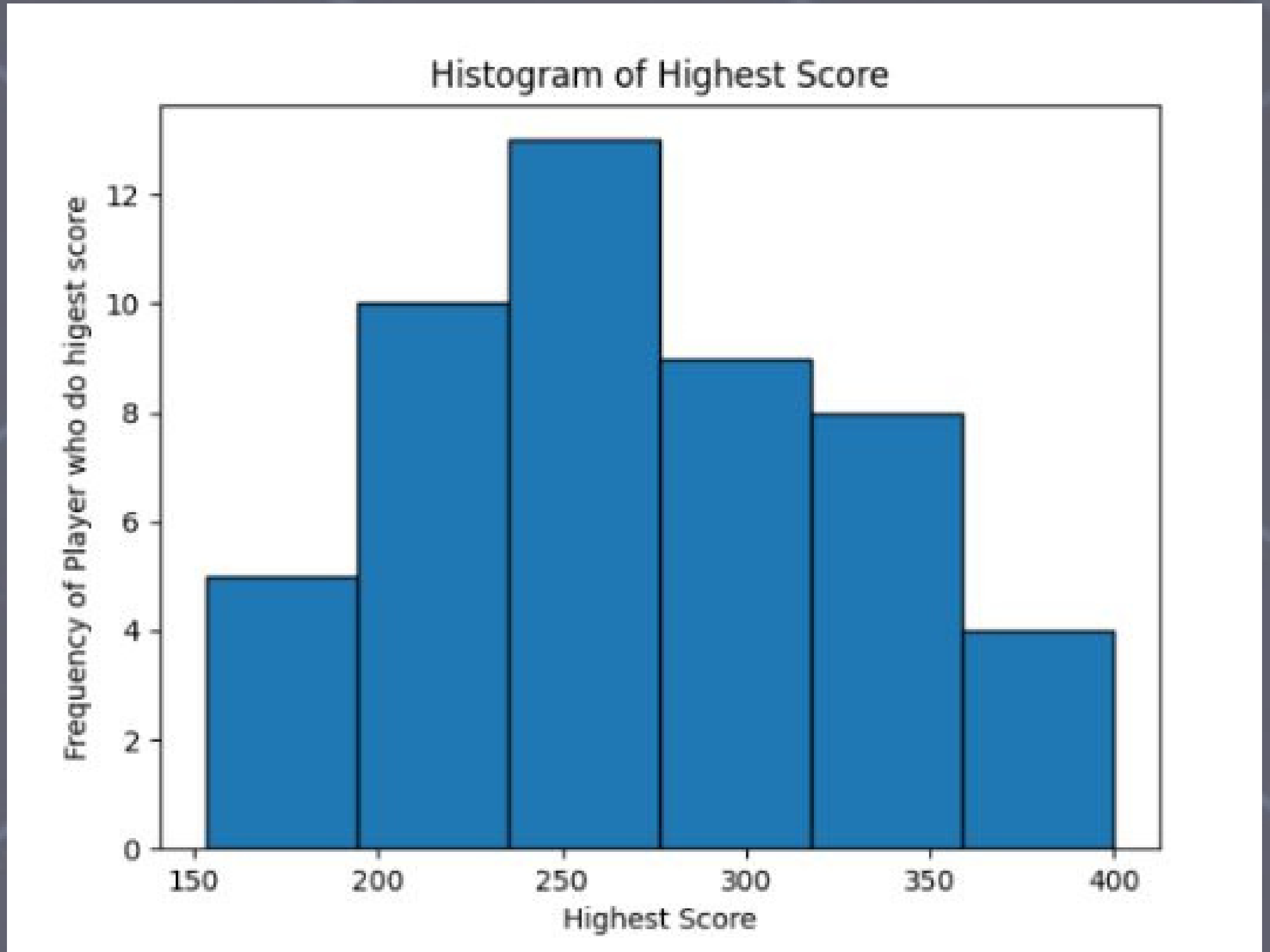
Data Visualization

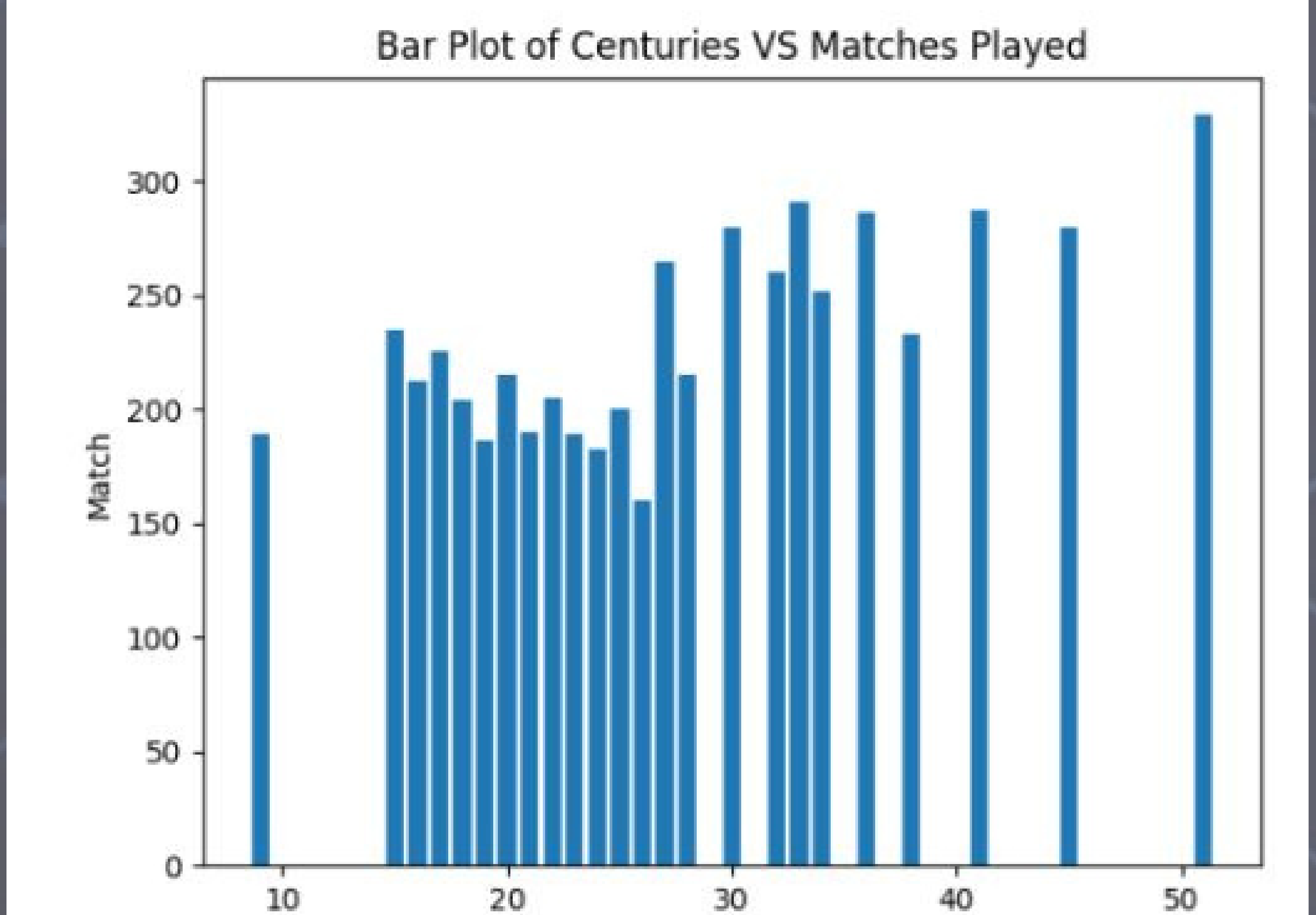
```
#Bar Plot of Centuries VS Matches Played
cen_x = data['Centuries']
mat_y = data['Matches Played']

plt.bar(cen_x,mat_y)
plt.title("Bar Plot of Centuries VS Matches Played")
plt.xlabel("Centuries")
plt.ylabel("Match")
plt.show()

#Histogram of Upto Year Played

up_his=data['Upto year played']
plt.hist(up_his,bins=5,edgecolor='black')
plt.xlabel('Upto year played')
plt.ylabel('Frequency Of Player')
plt.title('Histogram Of Upto Year Played')
plt.show()
```

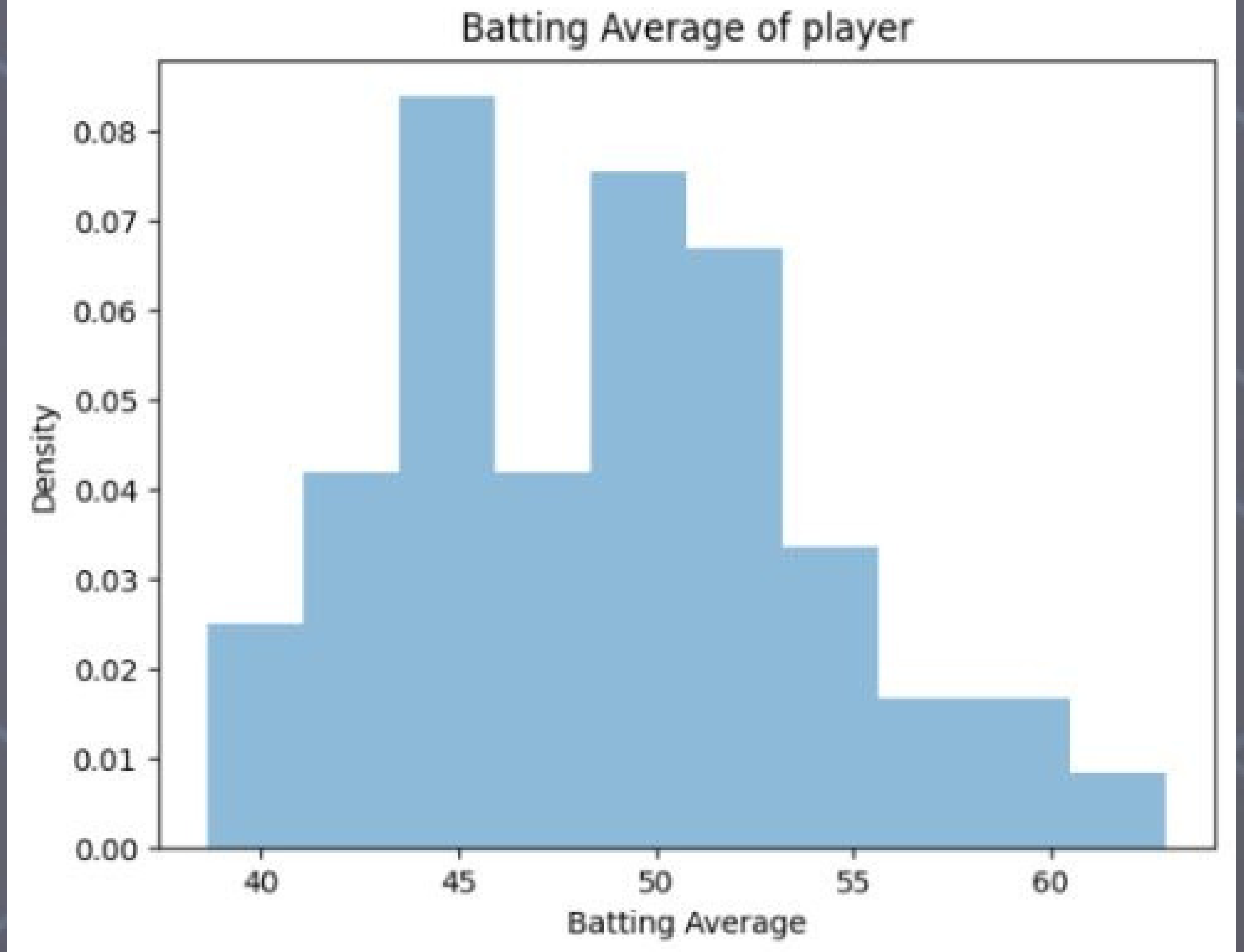
```
#Density Plot of Batting Average
Bat_avg=data['Batting Average']
dens1=np.random.normal(Bat_avg)
plt.hist(dens1,density=True,bins=10,alpha=0.5)
plt.xlabel('Batting Average')
plt.ylabel('Density')
plt.title('Batting Average of player')
plt.show()
```

Data Visualization









```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.axes as ax
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.cluster import KMeans
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.axes as ax
```

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.cluster import KMeans

data=pd.read_csv("/content/sample_data/ICC Test Batting Figures.csv",encoding=
'unicode_escape')
data1={'x':data["Runs Scored"], 'y':data["Fifties"]}
data=pd.DataFrame(data1, columns=['x','y'])

km= KMeans(n_clusters=3).fit(data)
centroids=km.cluster_centers_

plt.xlabel('Runs Scored')
plt.ylabel('Fifties')
plt.scatter(data['x'],data['y'],c=km.labels_.astype(float),s=60, alpha=1)
plt.scatter(centroids[:,0],centroids[:,1],c='red',s=190)
plt.show()

data=pd.read_csv("/content/sample_data/ICC    Test    Batting    Figures.csv",encoding=
'unicode_escape')
df = pd.DataFrame(data)

# Selecting the features for clustering

feat= df[['Batting Average','Matches Played']]

# Specifying the number of clusters
n_clusters = 4

# Fitting the K-means model
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
kmeans.fit(feat)
```

```
# Adding the cluster labels to the DataFrame
df['Cluster'] = kmeans.labels_

# Visualizing the clusters
plt.scatter(feat['Matches Played'], feat['Batting Average'], c=df['Cluster'], cmap='viridis')
plt.xlabel('Matches Played')
plt.ylabel('Batting Average')
plt.title('K-means Clustering')
plt.show()

data=pd.read_csv("/content/sample_data/ICC Test Batting Figures.csv",encoding=
'unicode_escape')
df = pd.DataFrame(data)

# Selecting the features for clustering


feat= df[['Fifties','Runs Scored']]
```

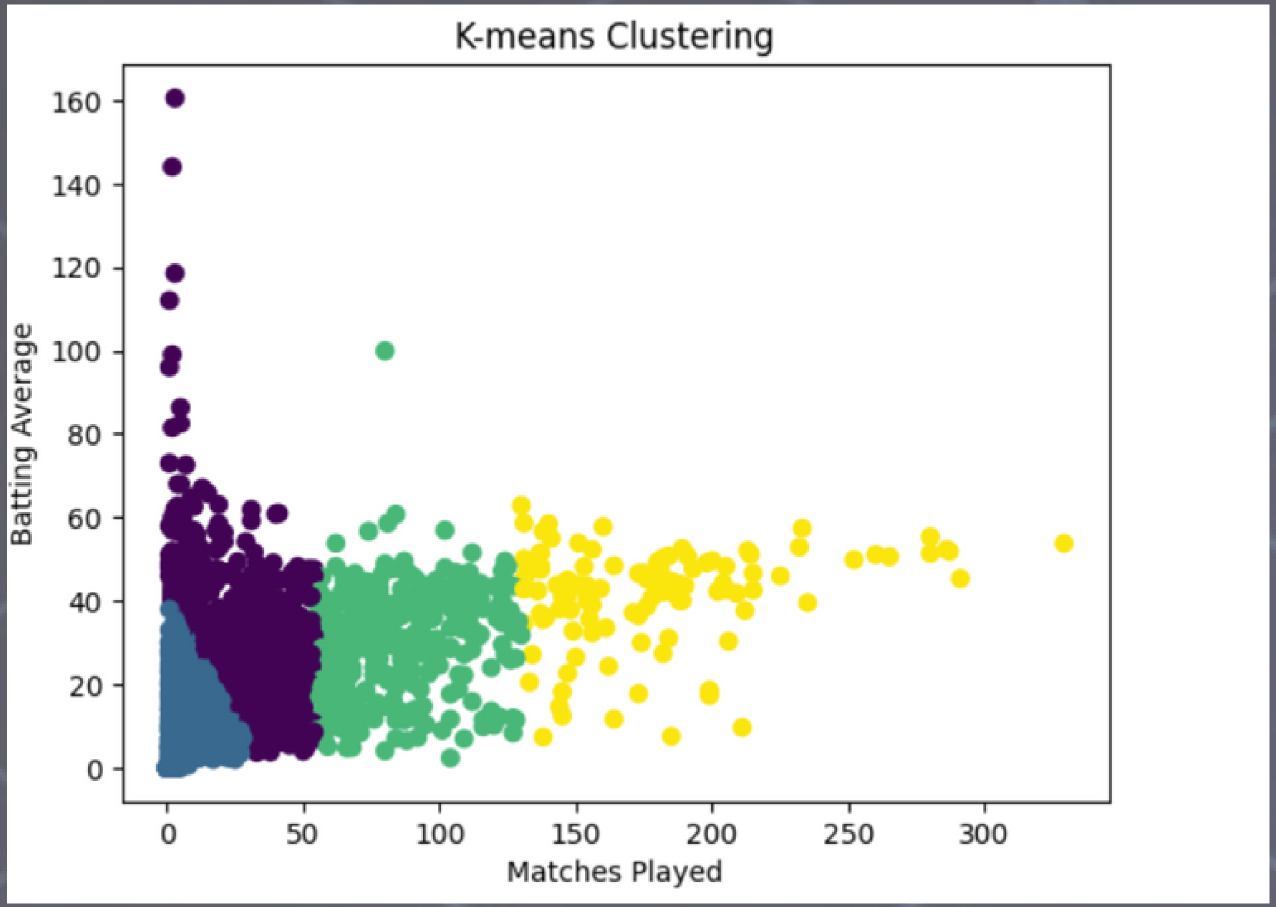
```
# Specifying the number of clusters
n_clusters = 4

# Fitting the K-means model
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
kmeans.fit(feat)

# Adding the cluster labels to the DataFrame
df['Cluster'] = kmeans.labels_

# Visualizing the clusters
plt.scatter(feat['Runs Scored'], feat['Fifties'], c=df['Cluster'], cmap='viridis')
plt.xlabel('Matches Played')
plt.ylabel('Batting Average')
plt.title('K-means Clustering')
plt.show()
```

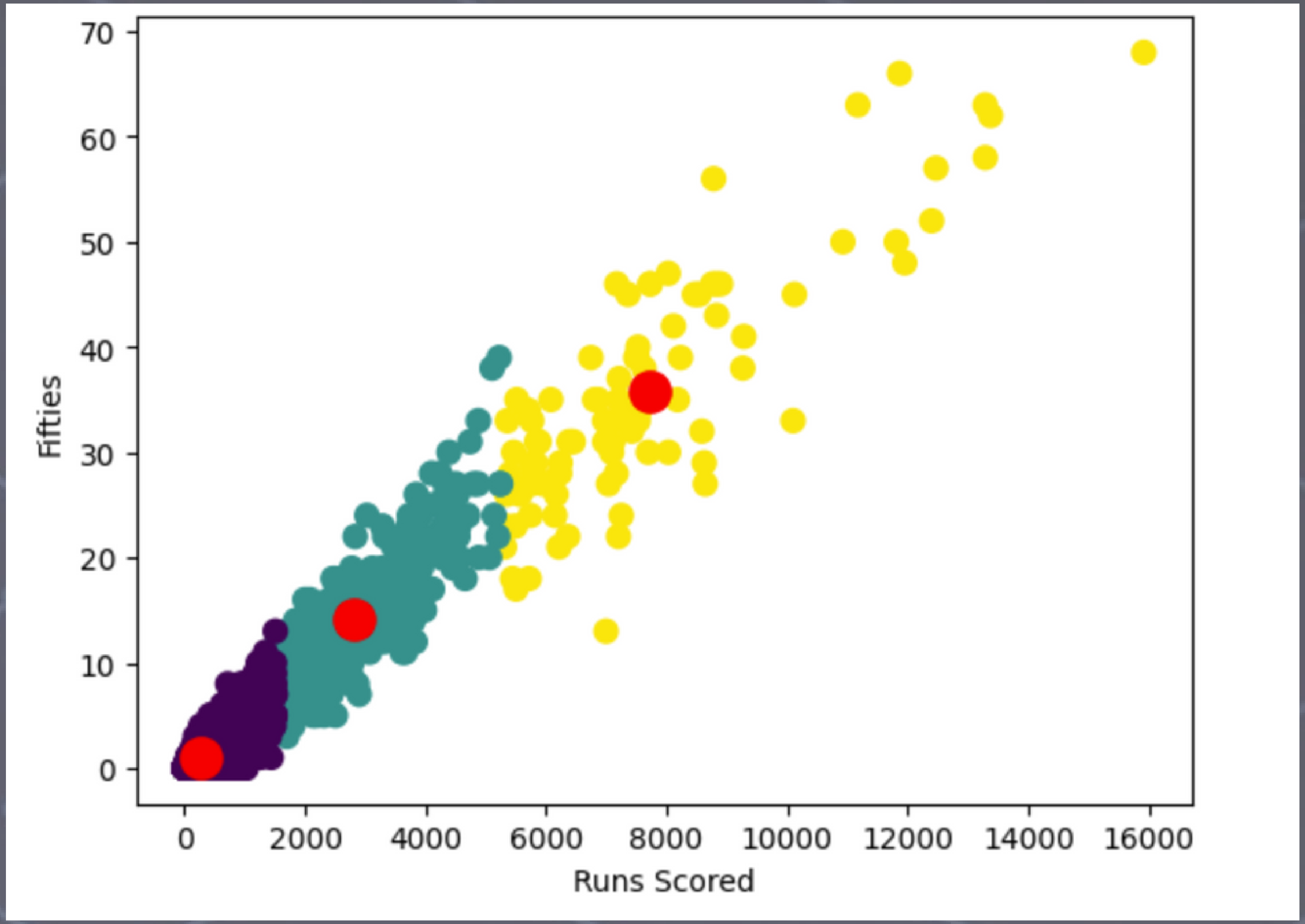
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

data=pd.read_csv("/content/sample_data/ICC Test Batting Figures.csv",encoding=
'unicode_escape')
x=np.array(data['Matches Played']).reshape(-1,1)
y=np.array(data['Fifties'])

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/3,random_state=0)
regressor= LinearRegression()
regressor.fit(x_train,y_train)
```

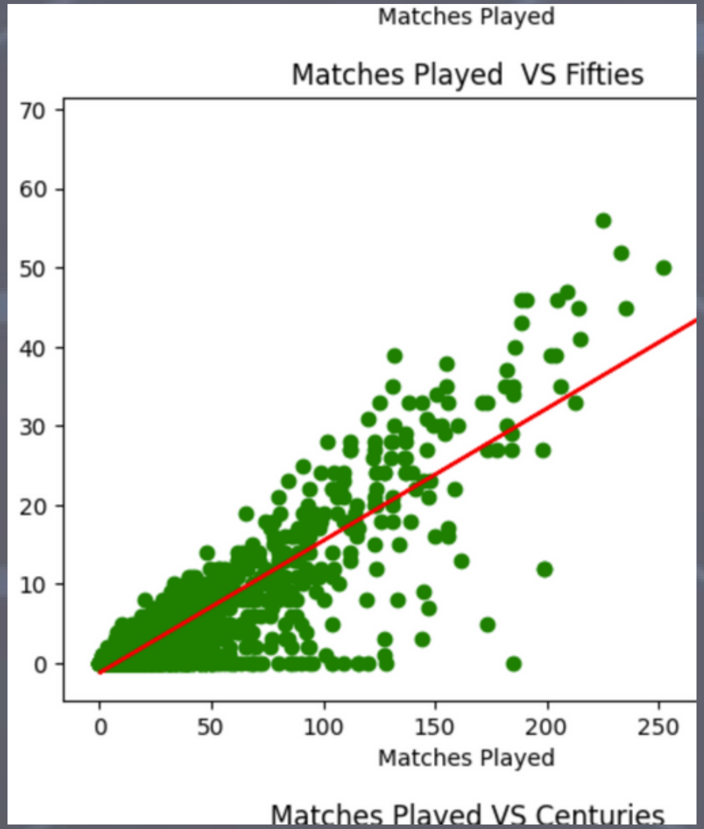
```
y_pred=regressor.predict(x_test)
x_pred=regressor.predict(x_train)

plt.scatter(x_train, y_train, color="green")
plt.plot(x_train, x_pred, color="red")
plt.title("Matches Played VS Fifties")
plt.xlabel("Matches Played")
plt.ylabel("Fifties")
plt.show()

data=pd.read_csv("/content/sample_data/ICC Test Batting Figures.csv",encoding=
'unicode_escape')

x=np.array(data['Matches Played']).reshape(-1,1)
y=np.array(data['Centuries']}
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/3,random_state=0)
regressor= LinearRegression()
regressor.fit(x_train,y_train)

y_pred=regressor.predict(x_test)
x_pred=regressor.predict(x_train)

plt.scatter(x_train, y_train, color="green")
plt.plot(x_train, x_pred, color="red")
plt.title("Matches Played VS Centuries")
plt.xlabel("Matches Played")
plt.ylabel("Centuries")
plt.show()

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

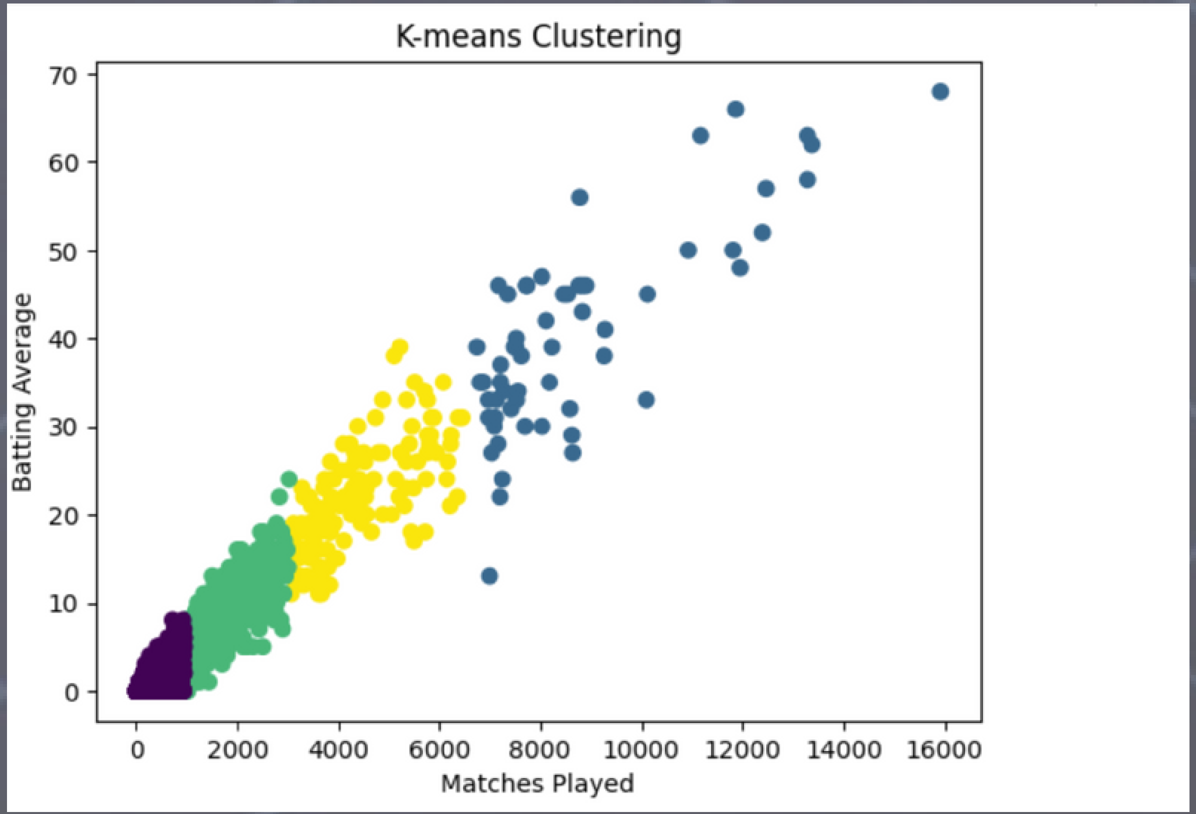
```
# Sample dataset
X = [[5, 3], [10, 15], [15, 12], [24, 10], [30, 45], [85, 70]]
y = [0, 1, 1, 0, 1, 0] # Labels for the samples

# Splitting the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Creating a KNN classifier object
knn = KNeighborsClassifier(n_neighbors=3)

# Training the classifier
knn.fit(X_train, y_train)

# Predicting labels for the test set
y_pred = knn.predict(X_test)
```

```
# Calculating the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```


K-means Clustering
Batting Average
Matches Played


Fifties
Runs Scored


Matches Played
Matches Played VS Fifties
Matches Played
Matches Played VS Centuries


K-means Clustering
Batting Average
Matches Played