EDS ASSIGNMENT 06 & Minor Project

Prof. Pushpmala Shinde
Mam

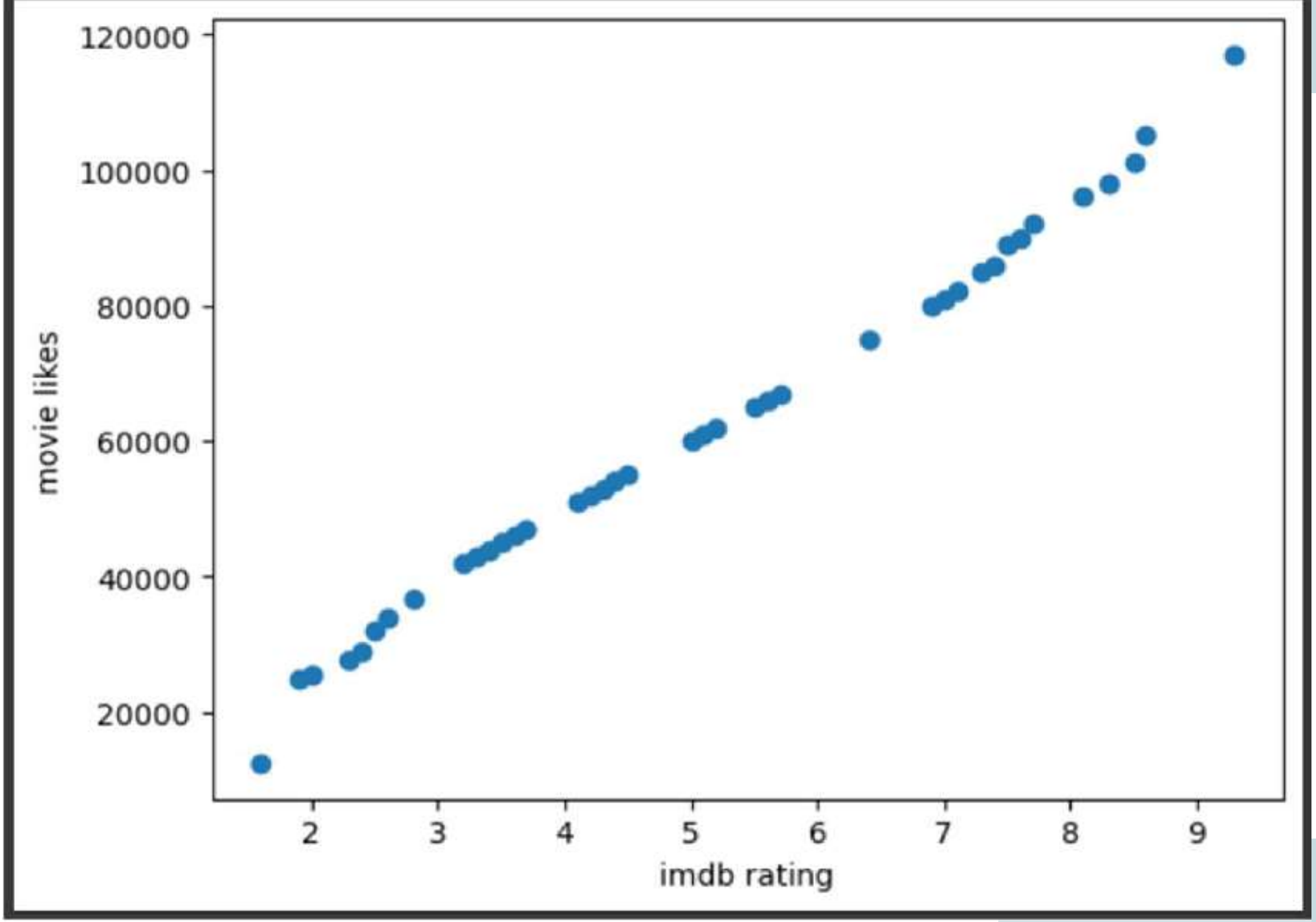
Project Members

1. Rohan Budgujar 502
2. Varun Balbudhe 503
3. Abhay Isal 519

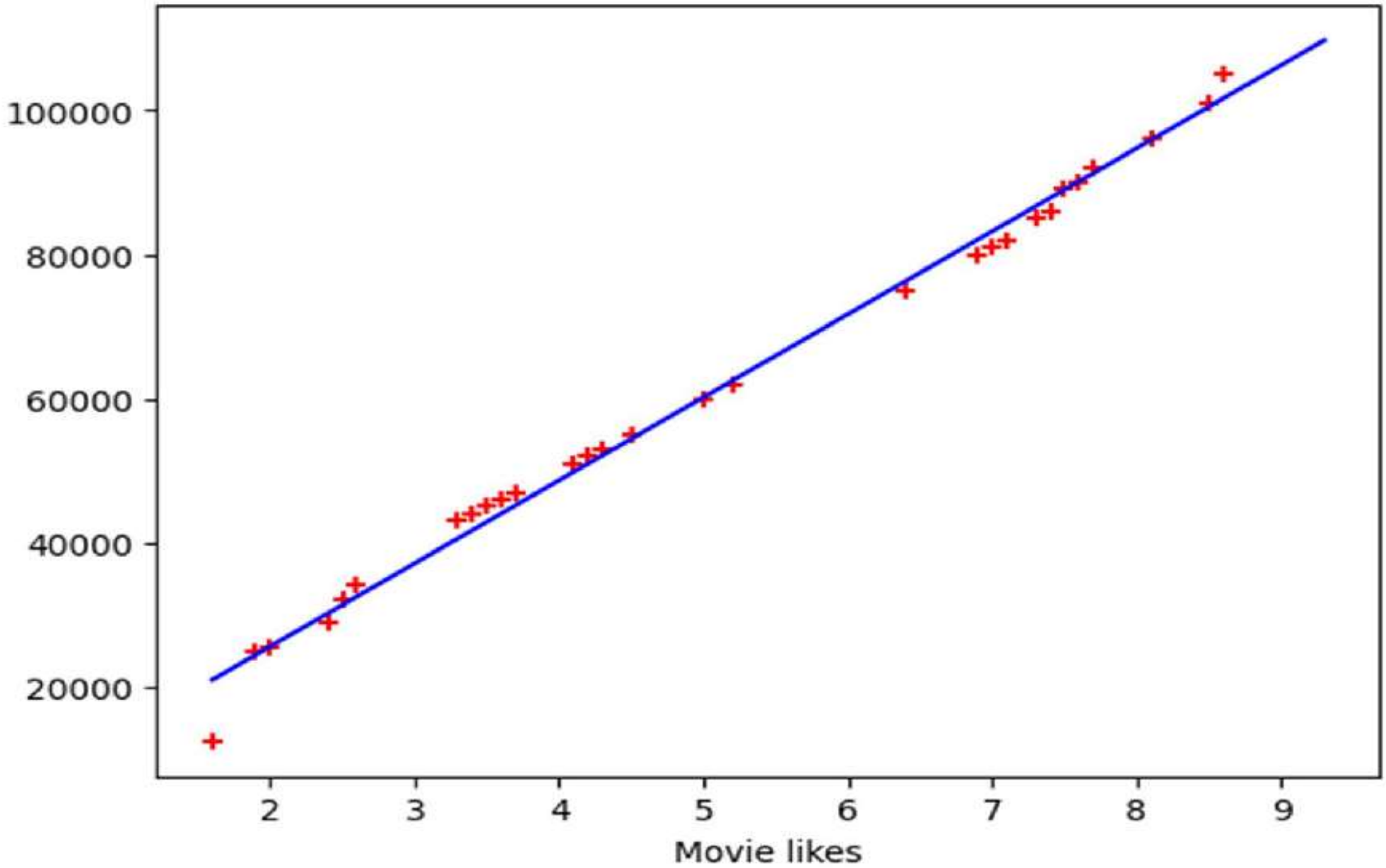
ESSENTIALS OF DATA SCIENCE

Using linear regression

```
1   import numpy as np
2   import pandas as pd
3   import matplotlib.pyplot as plt
4   from sklearn import linear_model
5   from sklearn.model_selection import train_test_split
6   df = pd.read_csv("/content/movie_data.csv")
7
8   #data cleaning  df.dropna(inplace=True)
9   df.reset_index(drop=True, inplace=True)
10  df1 = df.head(40)
11  # print(df1)
12  plt.scatter(df1['imdb_score'], df1['movie_likes'])
13  plt.xlabel('imdb rating')
14  plt.ylabel('movie likes')
```


120000
100000
80000
60000
40000
20000
movie likes
2
3
4
5
6
7
8
9
imdb rating

```
16  X = np.array(df1[['imdb_score']]).reshape(-1,1)
17  Y = np.array(df1[['movie_likes']]).reshape(-1,1)
18  X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.25)
19
20  # create linear regression object
21  reg = linear_model.LinearRegression()
22  reg.fit(X_train, Y_train)   #training the model
23  # predicting movie likes using the testing dataset on the trained model
24  reg.predict(X_test)
25  # ploting linear regression line
26  plt.scatter(X_train, Y_train, color='red', marker='+')
27  plt.xlabel('IMDB')
28  plt.xlabel('Movie likes')
29  plt.plot(df1['imdb_score'],
30  reg.predict(df1[['imdb_score']]),
31  color='blue')
```

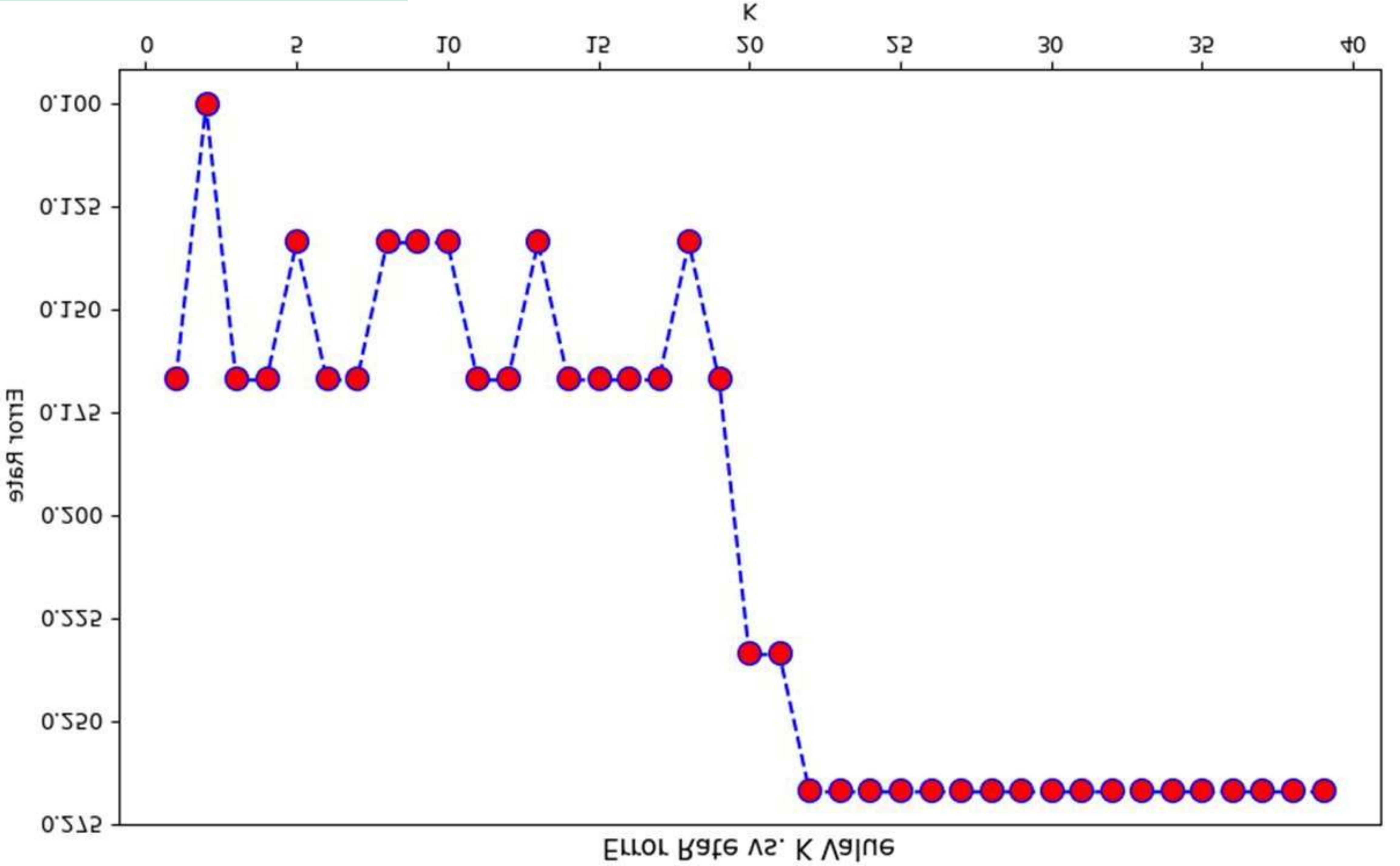

100000
80000
60000
40000
20000
2
3
4
5
6
7
8
9
Movie likes

Using KNN

```
1  import pandas as pd
2  import seaborn as sns
3  import matplotlib.pyplot as plt
4  import numpy as np
5  df = pd.read_csv("prostate.csv")
6  df.head()
7  from sklearn.preprocessing import StandardScaler
8  scaler = StandardScaler()
9  scaler.fit(df.drop('Target', axis=1))
10 scaled_features = scaler.transform(df.drop('Target',axis=1))
11
```

```
df_feat = pd.DataFrame(scaled_features,  columns=df.columns[:-1])
df_feat.head()
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
X_train, X_test,\
y_train, y_test = train_test_split(scaled_features,
df['Taregt'],  test_size=0.30)
# Remember that we are trying to come up   # with a model to predict whether
# someone will Target or not.   # We'll start with k = 1.
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)
# Predictions and Evaluations
#  Let's evaluate our KNN model !
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
error_rate = []
# Will take some time  for i in range(1, 40):
knn = KNeighborsClassifier(n_neighbors=i)
knn.fit(X_train, y_train)
pred_i = knn.predict(X_test)
error_rate.append(np.mean(pred_i != y_test))
```

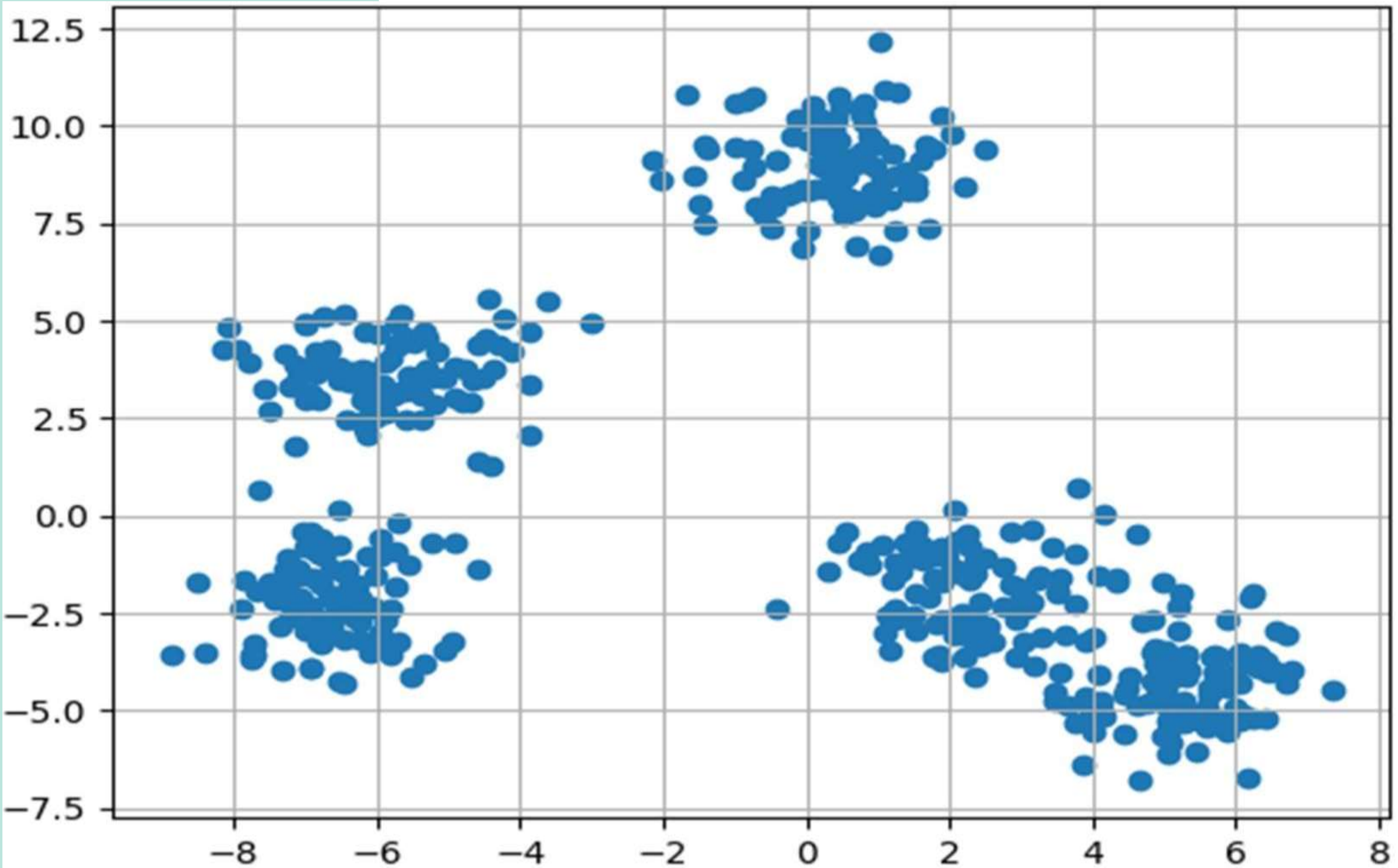
```
print(classification_report(y_test, pred))
error_rate = []
# Will take some time
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))

plt.figure(figsize=(10, 6))
plt.plot(range(1, 40), error_rate, color='blue',  linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)

plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
plt.show()
```


Error Rate vs. K Value
K
Error Rate
0
5
10
15
20
25
30
35
40
0.100
0.125
0.150
0.175
0.200
0.225
0.250
0.275

```
47  # FIRST A QUICK COMPARISON TO OUR ORIGINAL K = 1
48  knn = KNeighborsClassifier(n_neighbors = 1)
49
50  knn.fit(X_train, y_train)
51  pred = knn.predict(X_test)
52  print('WITH K = 1')
53  print('Confusion Matrix')
54  print(confusion_matrix(y_test, pred))
55  print('Classification Report')
56  print(classification_report(y_test, pred))
57
58  # NOW WITH K = 10
59  knn = KNeighborsClassifier(n_neighbors = 10)
60  knn.fit(X_train, y_train)
61  pred = knn.predict(X_test)
62  print('WITH K = 10')
63  print('Confusion Matrix')
64  print(confusion_matrix(y_test, pred))
65  print('Classification Report')
66  print(classification_report(y_test, pred))
```

Using K Means

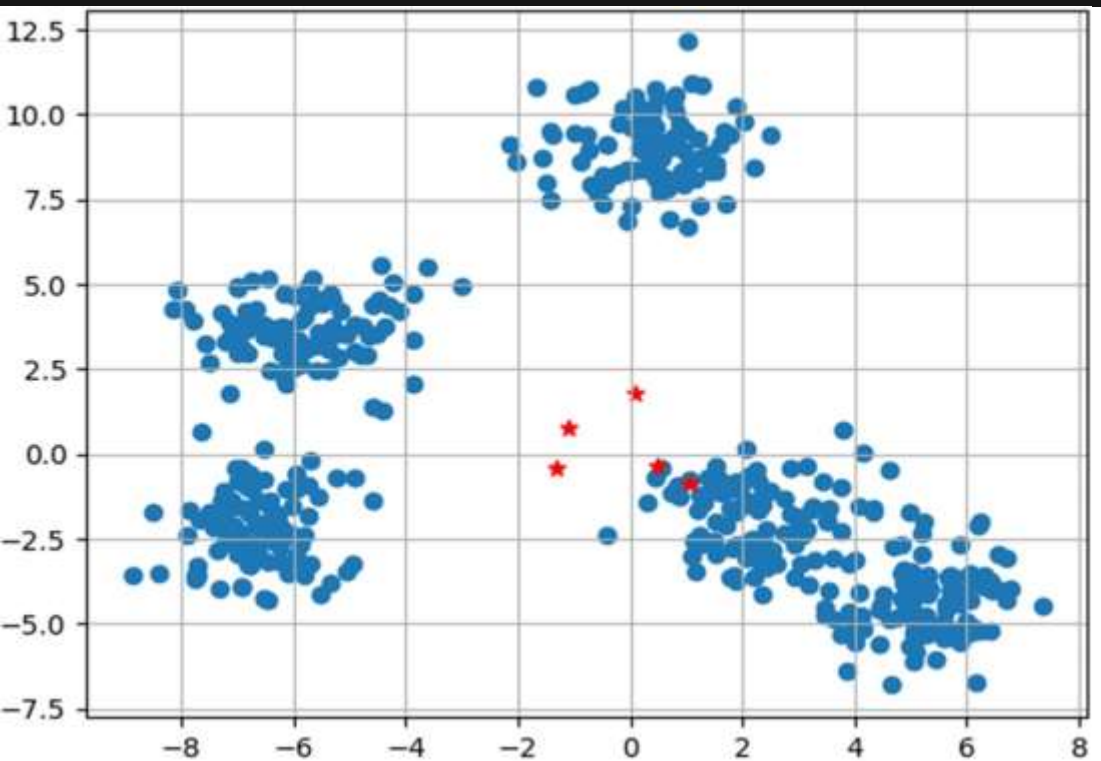
```
1  import matplotlib.pyplot as plt
2  from sklearn.datasets import make_blobs
3  X,y = make_blobs(n_samples = 500,n_features = 2,centers = 5,random_state = 23)
4  fig = plt.figure(0)
5  plt.grid(True)
6  plt.scatter(X[:,0],X[:,1])
7  plt.show()
8
```


12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−8
−6
−4
−2
0
2
4
6
8

```python
import numpy as np
clusters = {}
np.random.seed(23)

for idx in range(k):
    center = 2*(2*np.random.random((X.shape[1],))-1)
    points = []
    cluster = {
        'center' : center,
        'points' : []
}
clusters[idx] = cluster


clusters
```

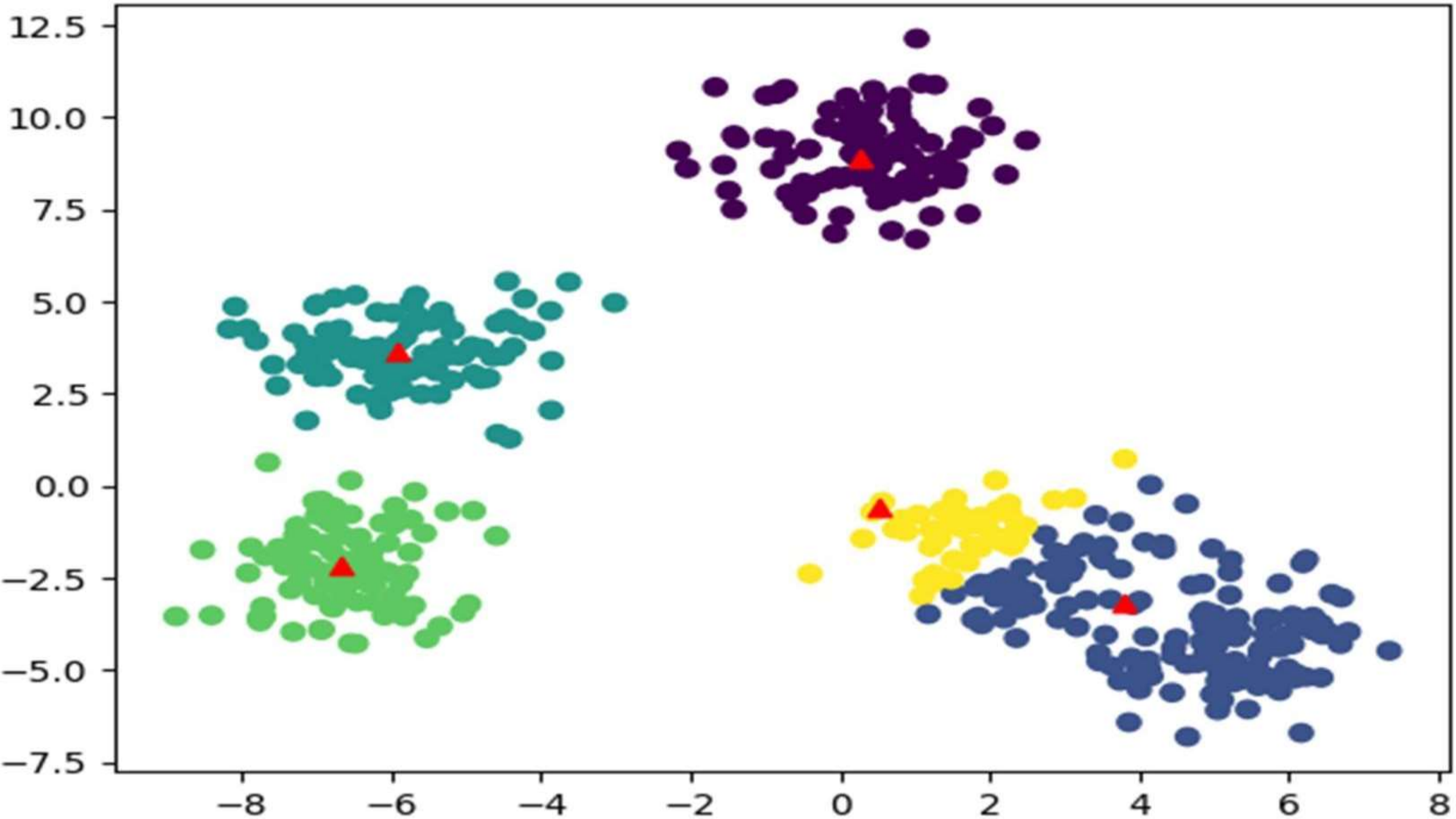
```
{0: {'center': array([0.06919154, 1.78785042]), 'points': []},
 1: {'center': array([ 1.06183904, -0.87041662]), 'points': []},
 2: {'center': array([-1.11581855,  0.74488834]), 'points': []},
 3: {'center': array([-1.33144319, -0.43023013]), 'points': []},
 4: {'center': array([ 0.47220939, -0.35227962]), 'points': []}}
```

```
23    clusters
24    plt.scatter(X[:,0],X[:,1])
25    plt.grid(True)
26    for i in clusters:
27        center = clusters[i]['center']
28    plt.scatter(center[0],center[1],marker = '*',c = 'red')
29    plt.show()
30
```



```python
#Implementing E step
def assign_clusters(X, clusters):
for idx in range(X.shape[0]):
    dist = []
    curr_x = X[idx]
    for i in range(k):
        dis = distance(curr_x,clusters[i]['center'])
        dist.append(dis)
        curr_cluster = np.argmin(dist)
        clusters[curr_cluster]['points'].append(curr_x)
        return clusters

#Implementing the M-Step
def update_clusters(X, clusters):
for i in range(k):
    points = np.array(clusters[i]['points'])
    if points.shape[0] > 0:
        new_center = points.mean(axis =0)
        clusters[i]['center'] = new_center

clusters[i]['points'] = []
return clusters

def pred_cluster(X, clusters):
    pred = []
    for i in range(X.shape[0]):
        dist = []
        for j in range(k):  dist.append(distance(X[i],clusters[j]['center']))
        pred.append(np.argmin(dist))
        return pred
```

```
clusters = assign_clusters(X,clusters)
clusters = update_clusters(X,clusters)
pred = pred_cluster(X,clusters)

plt.scatter(X[:,0],X[:,1],c = pred)
for i in clusters:
    center = clusters[i]['center']
    plt.scatter(center[0],center[1],marker = '^',c = 'red')
    plt.show()
```


12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−8
−6
−4
−2
0
2
4
6
8