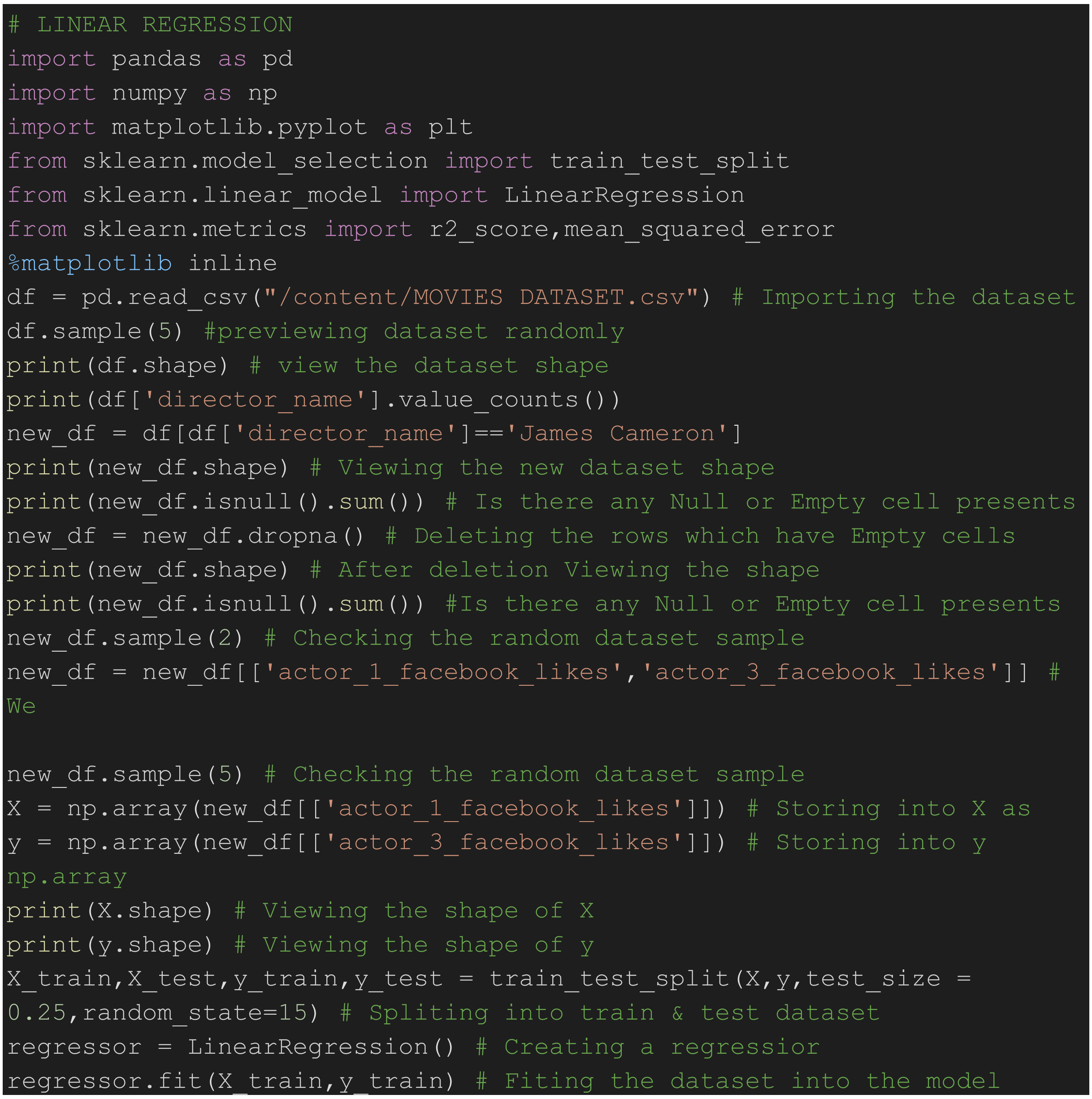
Assignment no. 6

Group no: 04

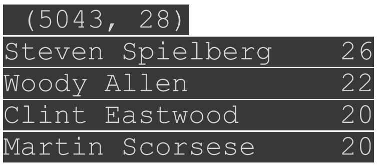
204 Aryan Meshram

210 Shreya Borle

212 Snehal Chavan



```
# LINEAR REGRESSION
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error
%matplotlib inline
df = pd.read_csv("/content/MOVIES DATASET.csv") # Importing the dataset
df.sample(5) #previewing dataset randomly
print(df.shape) # view the dataset shape
print(df['director_name'].value_counts())
new_df = df[df['director_name']=='James Cameron']
print(new_df.shape) # Viewing the new dataset shape
print(new_df.isnull().sum()) # Is there any Null or Empty cell presents
new_df = new_df.dropna() # Deleting the rows which have Empty cells
print(new_df.shape) # After deletion Viewing the shape
print(new_df.isnull().sum()) #Is there any Null or Empty cell presents
new_df.sample(2) # Checking the random dataset sample
new_df = new_df[['actor_1_facebook_likes','actor_3_facebook_likes']] #
We

new_df.sample(5) # Checking the random dataset sample
X = np.array(new_df[['actor_1_facebook_likes']]) # Storing into X as
y = np.array(new_df[['actor_3_facebook_likes']]) # Storing into y
np.array
print(X.shape) # Viewing the shape of X
print(y.shape) # Viewing the shape of y
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size =
0.25,random_state=15) # Spliting into train & test dataset
regressor = LinearRegression() # Creating a regressior
regressor.fit(X_train,y_train) # Fiting the dataset into the model
```

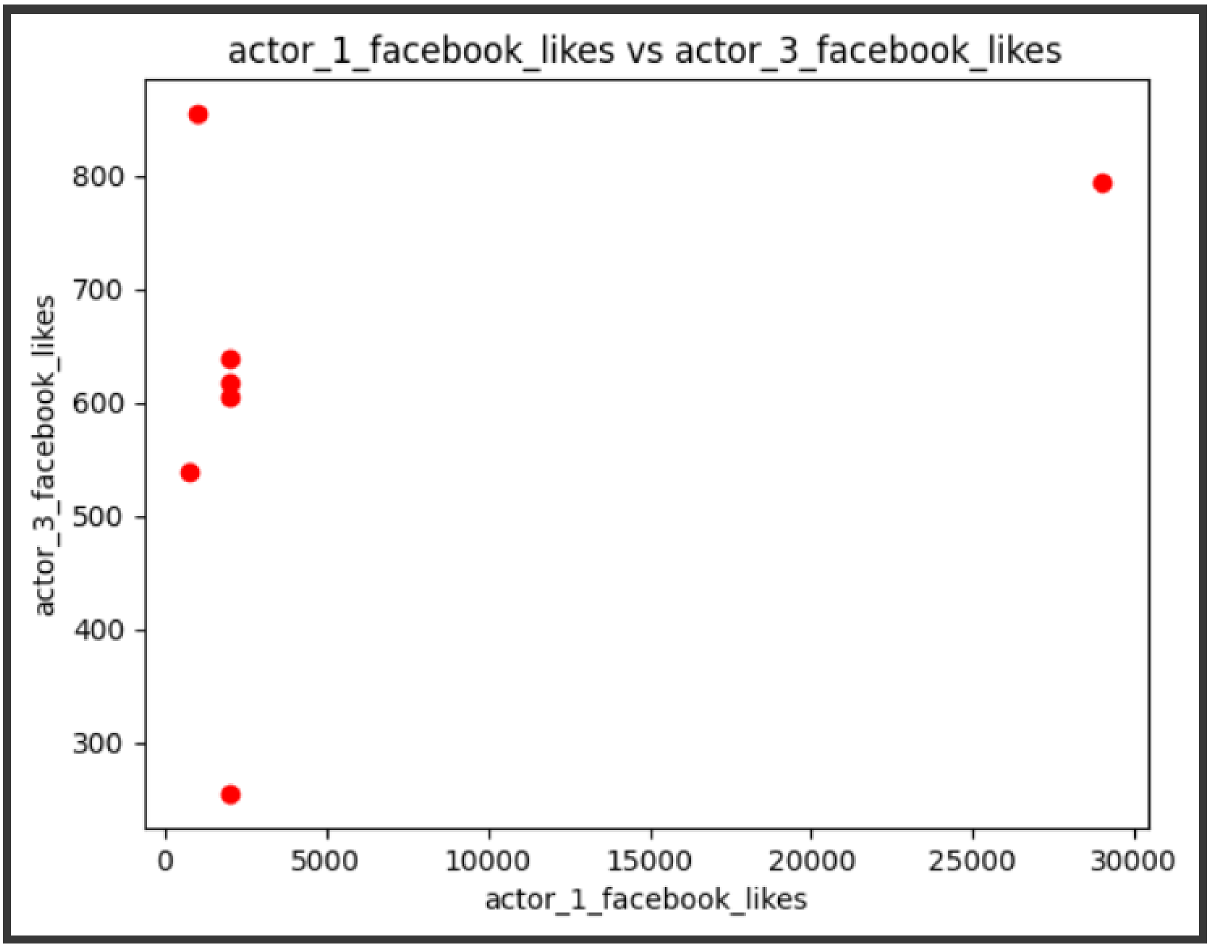


```
 (5043, 28)
Steven Spielberg    26
Woody Allen         22
Clint Eastwood      20
Martin Scorsese     20
```

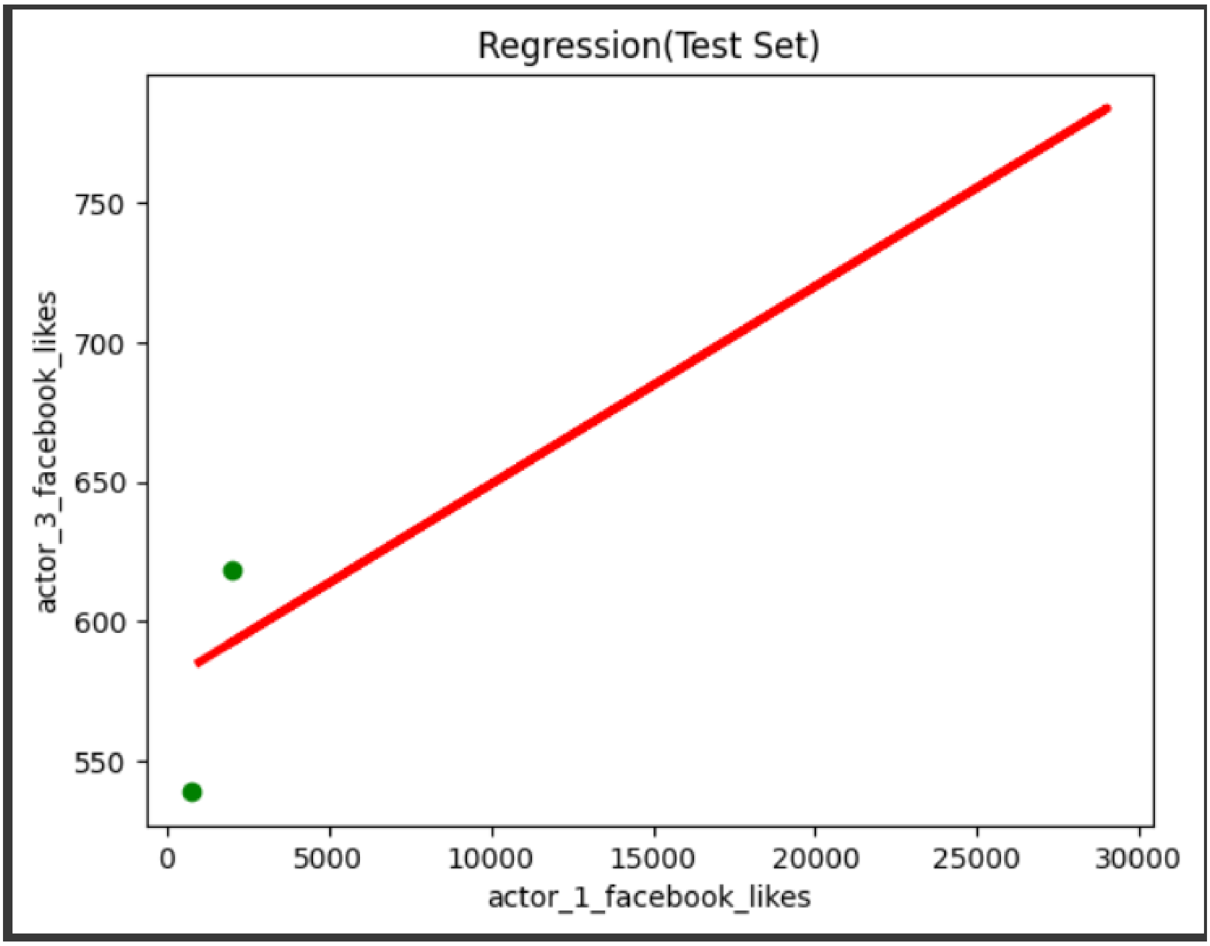
```
Ridley Scott         17
                     ..
John Crowley          1
Rob Pritts            1
David S. Ward         1
R.J. Cutler           1
Daniel Hsia           1
Name: director_name, Length: 2398, dtype: int64
(7, 28)
color                        0
director_name                0
num_critic_for_reviews       0
duration                     0
director_facebook_likes      0
actor_3_facebook_likes       0
actor_2_name                 0
actor_1_facebook_likes       0
gross                        0
genres                       0
actor_1_name                 0
movie_title                  0
num_voted_users              0
cast_total_facebook_likes    0
actor_3_name                 0
facenumber_in_poster         0
plot_keywords                0
movie_imdb_link              0
num_user_for_reviews         0
language                     0
country                      0
content_rating               0
budget                       0
title_year                   0
actor_2_facebook_likes       0
imdb_score                   0
aspect_ratio                 0
movie_facebook_likes         0
dtype: int64
(7, 28)
color                        0
director_name                0
num_critic_for_reviews       0
duration                     0
director_facebook_likes      0
actor_3_facebook_likes       0
actor_2_name                 0
actor_1_facebook_likes       0
gross                        0
genres                       0
actor_1_name                 0
movie_title                  0
num_voted_users              0
cast_total_facebook_likes    0
actor_3_name                 0
facenumber_in_poster         0
plot_keywords                0
movie_imdb_link              0
num_user_for_reviews         0
```

```
language                  0
country                   0
content_rating            0
budget                    0
title_year                0
actor_2_facebook_likes    0
imdb_score                0
aspect_ratio              0
movie_facebook_likes      0
dtype: int64
(7, 1)
(7, 1)
```

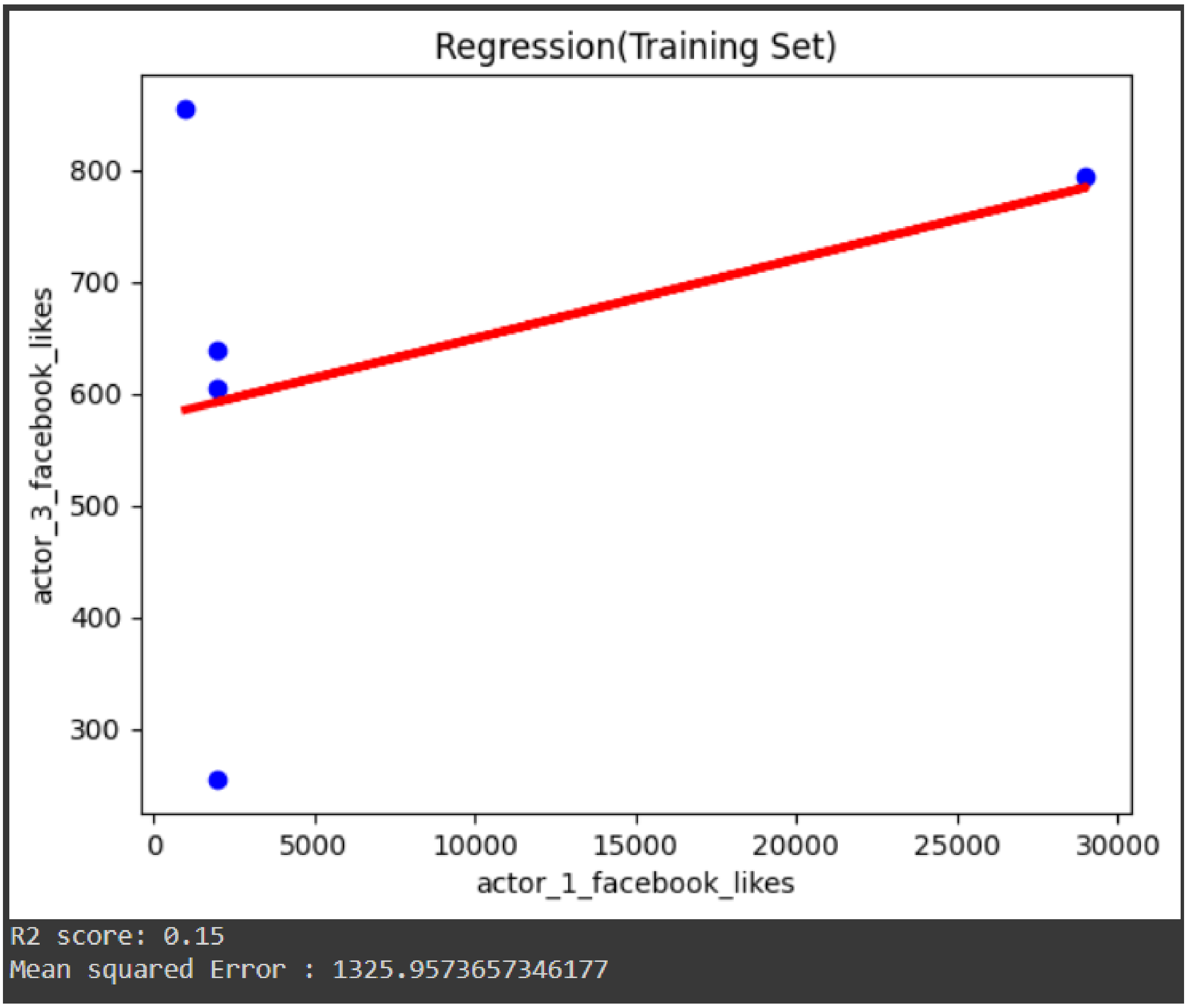
```
plt.scatter(X,y,color="red") # Plot a graph X vs y
plt.title('actor_1_facebook_likes vs actor_3_facebook_likes')
plt.xlabel('actor_1_facebook_likes')
plt.ylabel('actor_3_facebook_likes')
plt.show()
```



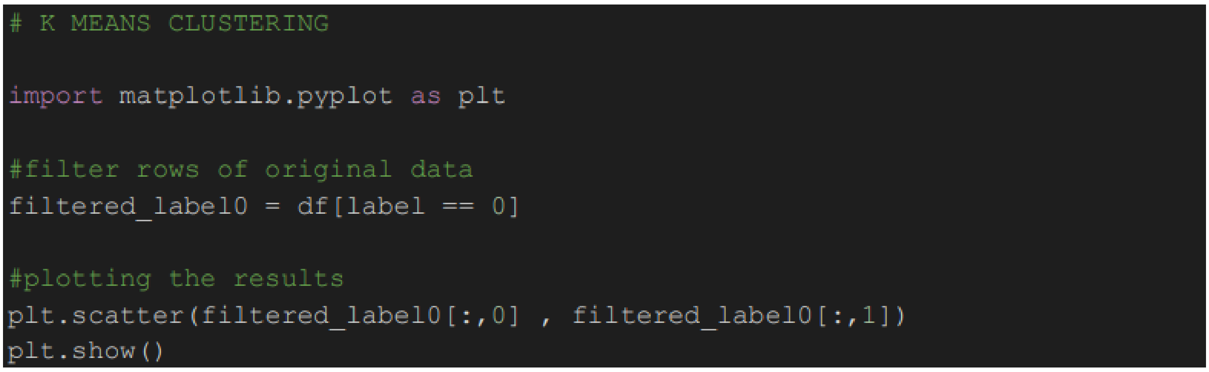
```
plt.scatter(X_test,y_test,color="green") # Plot a graph with X_test vs
y_test
plt.plot(X_train,regressor.predict(X_train),color="red",linewidth=3) #

plt.title('Regression(Test Set)')
plt.xlabel('actor_1_facebook_likes')
plt.ylabel('actor_3_facebook_likes')
plt.show()
```



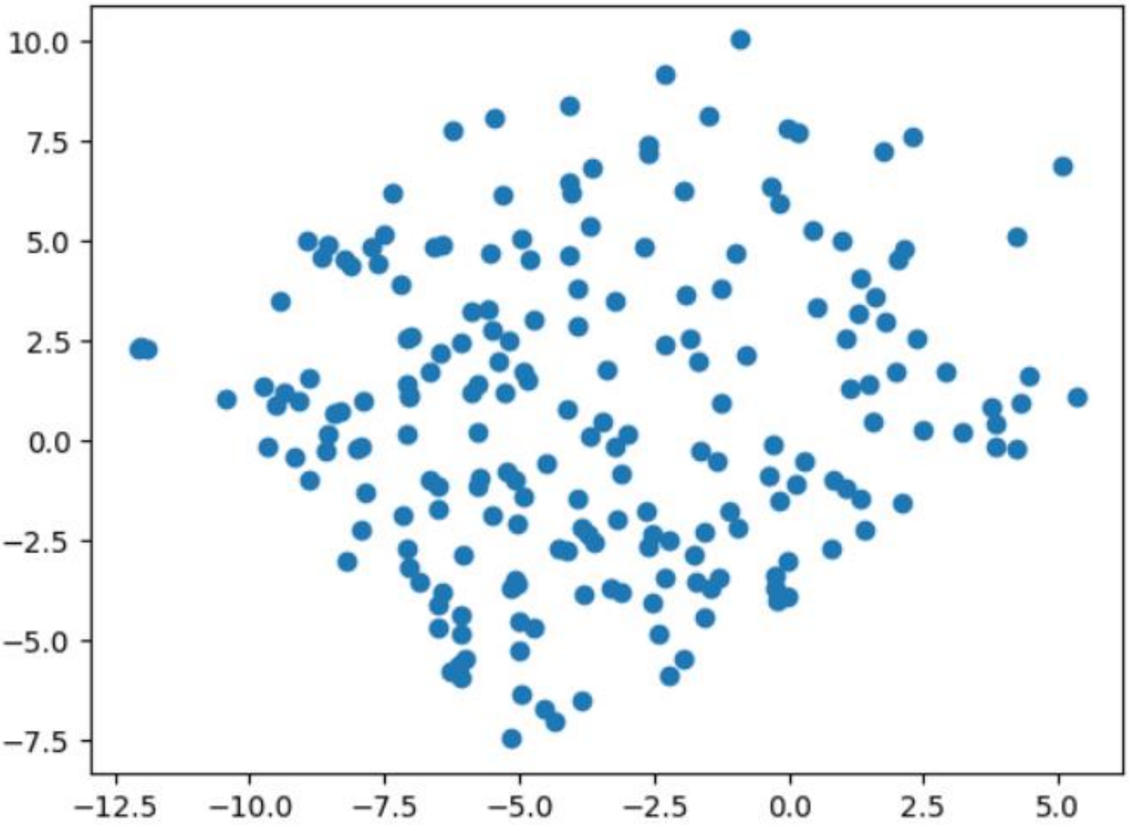
```
plt.scatter(X_train,y_train,color="blue")
# Plot a graph with X_train vs y_train
plt.plot(X_train,regressor.predict(X_train),color="red",linewidth=3)
# Regressior line showing
plt.title('Regression(Training Set)')
plt.xlabel('actor_1_facebook_likes')
plt.ylabel('actor_3_facebook_likes')
plt.show()
y_pred = regressor.predict(X_test)
print('R2 score: %.2f' % r2_score(y_test,y_pred)) # Priniting R2 Score
print('Mean squared Error :',mean_squared_error(y_test,y_pred))
# Priniting the mean error
```



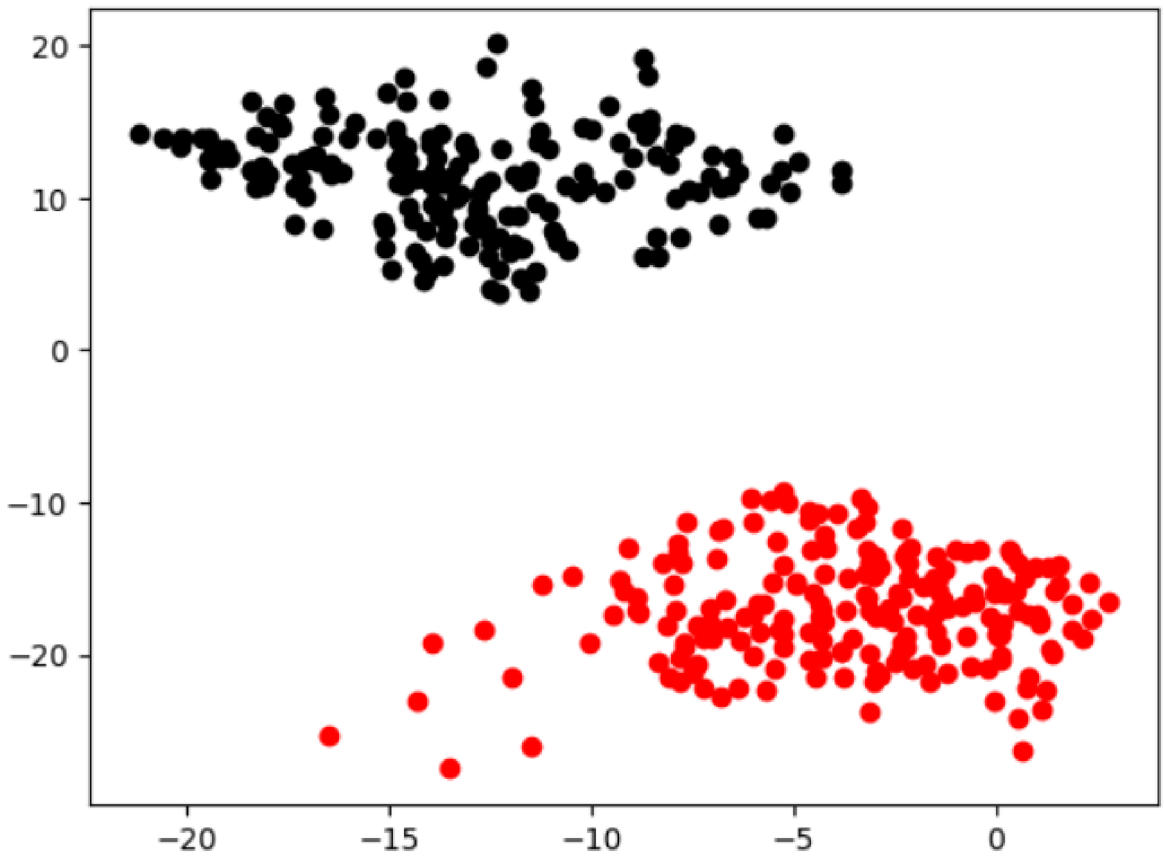
```
R2 score: 0.15
Mean squared Error : 1325.9573657346177
```

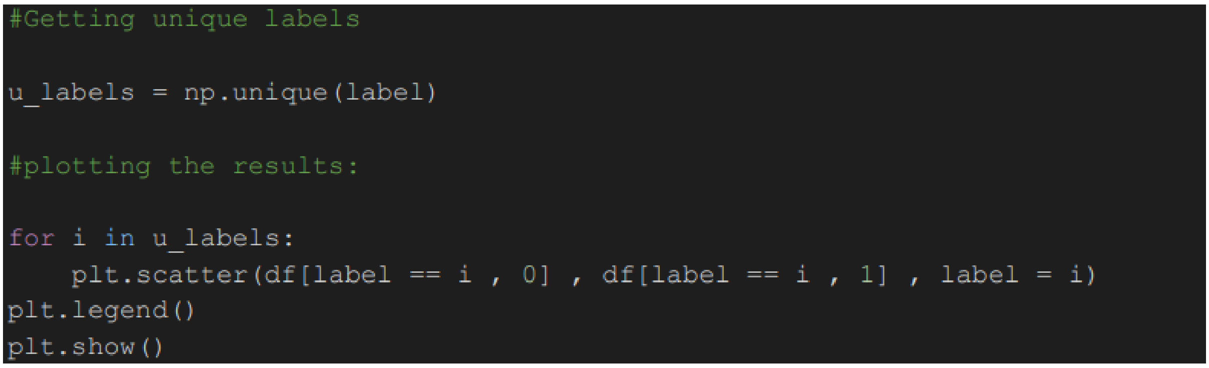


```
# K MEANS CLUSTERING

import matplotlib.pyplot as plt

#filter rows of original data
filtered_label0 = df[label == 0]

#plotting the results
plt.scatter(filtered_label0[:,0] , filtered_label0[:,1])
plt.show()
```

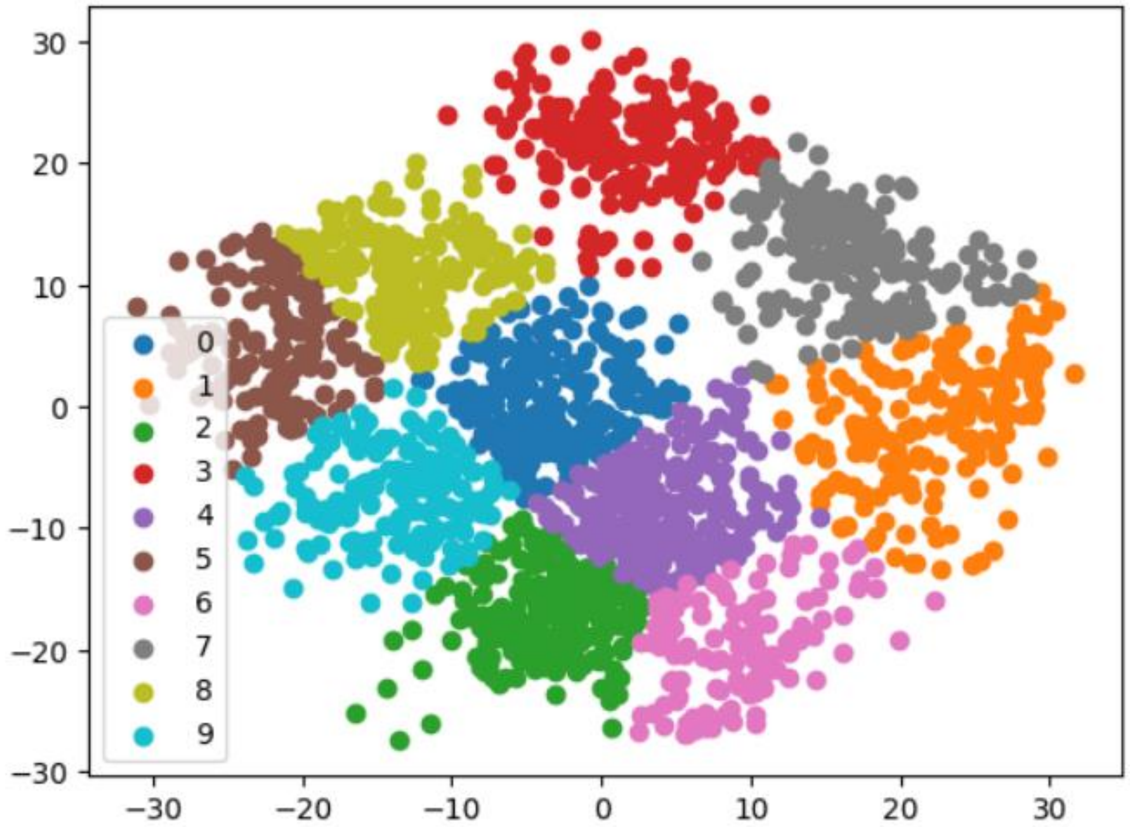


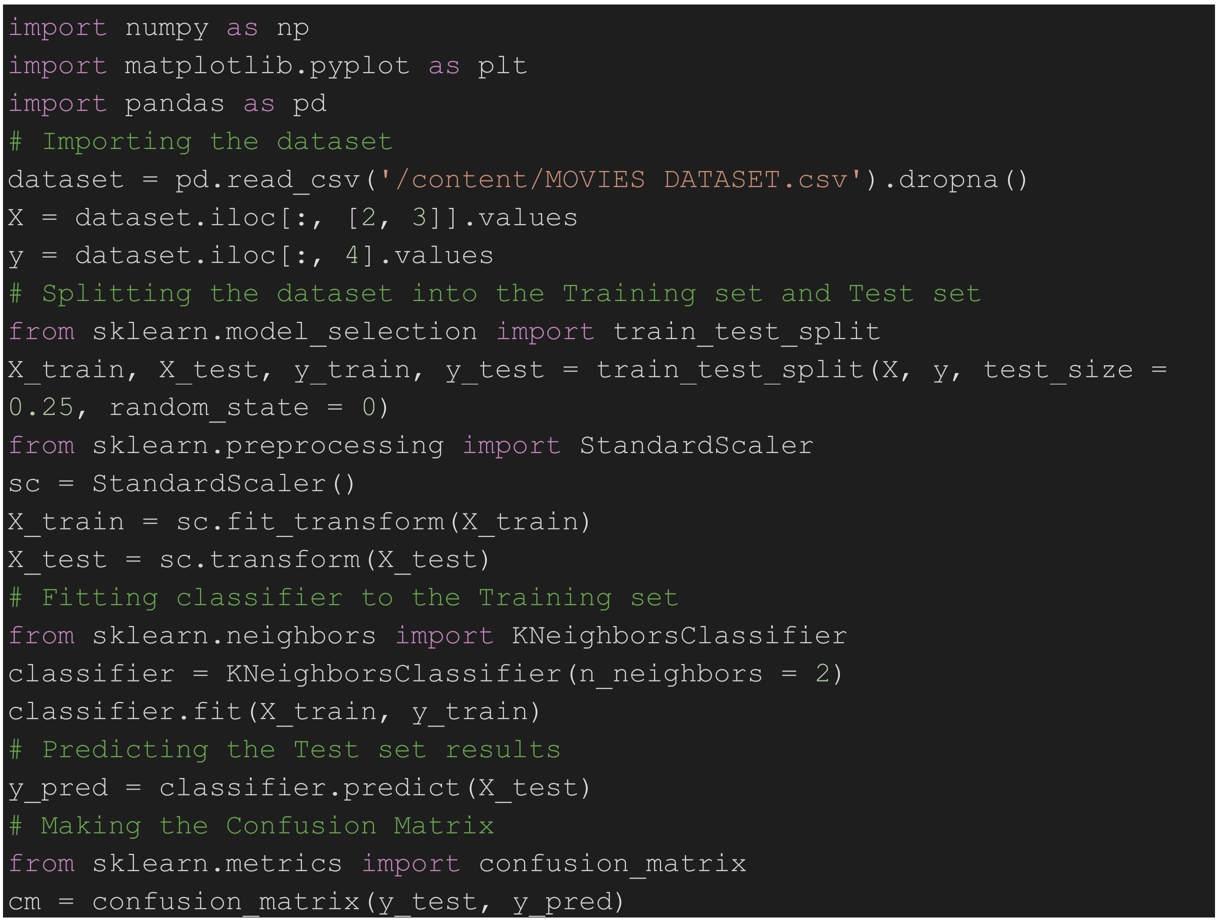
```
#filter rows of original data
filtered_label2 = df[label == 2]
filtered_label8 = df[label == 8]
#Plotting the results
plt.scatter(filtered_label2[:,0] , filtered_label2[:,1] , color =
'red')
plt.scatter(filtered_label8[:,0] , filtered_label8[:,1] , color =
'black')
plt.show()
```



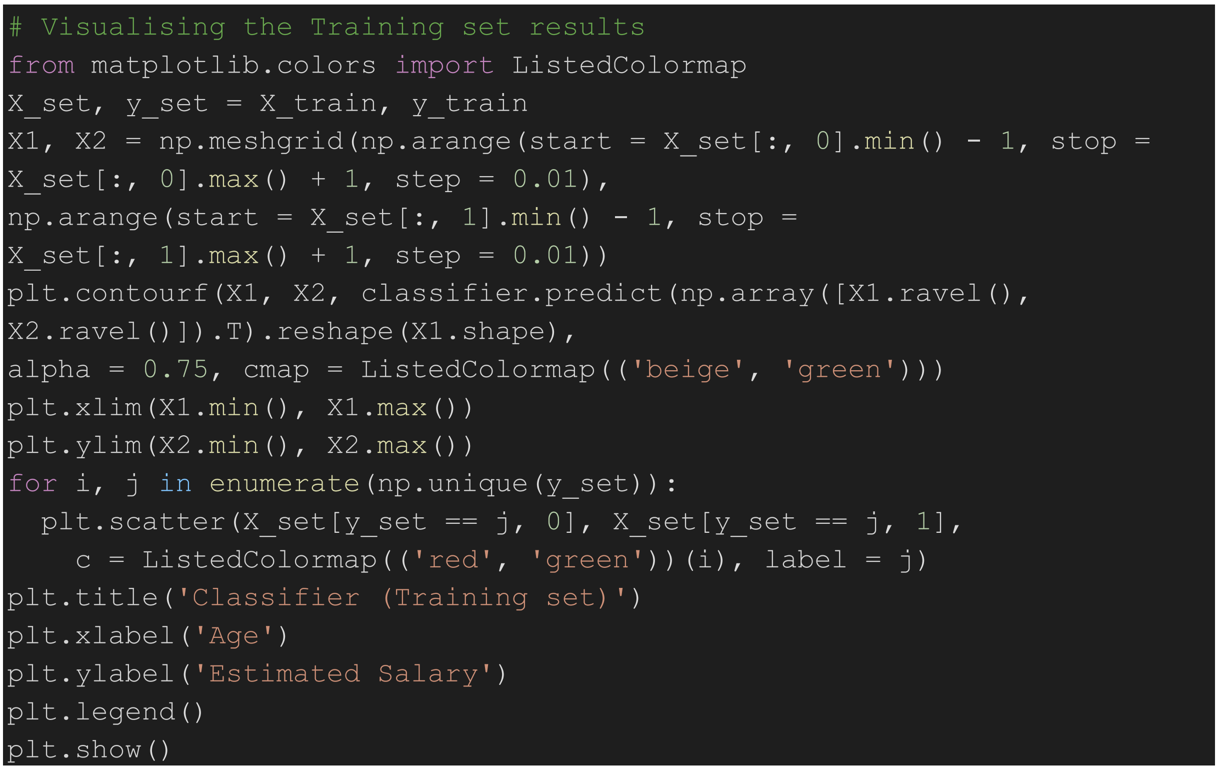


```
#Getting unique labels

u_labels = np.unique(label)

#plotting the results:

for i in u_labels:
    plt.scatter(df[label == i , 0] , df[label == i , 1] , label = i)
plt.legend()
plt.show()
```

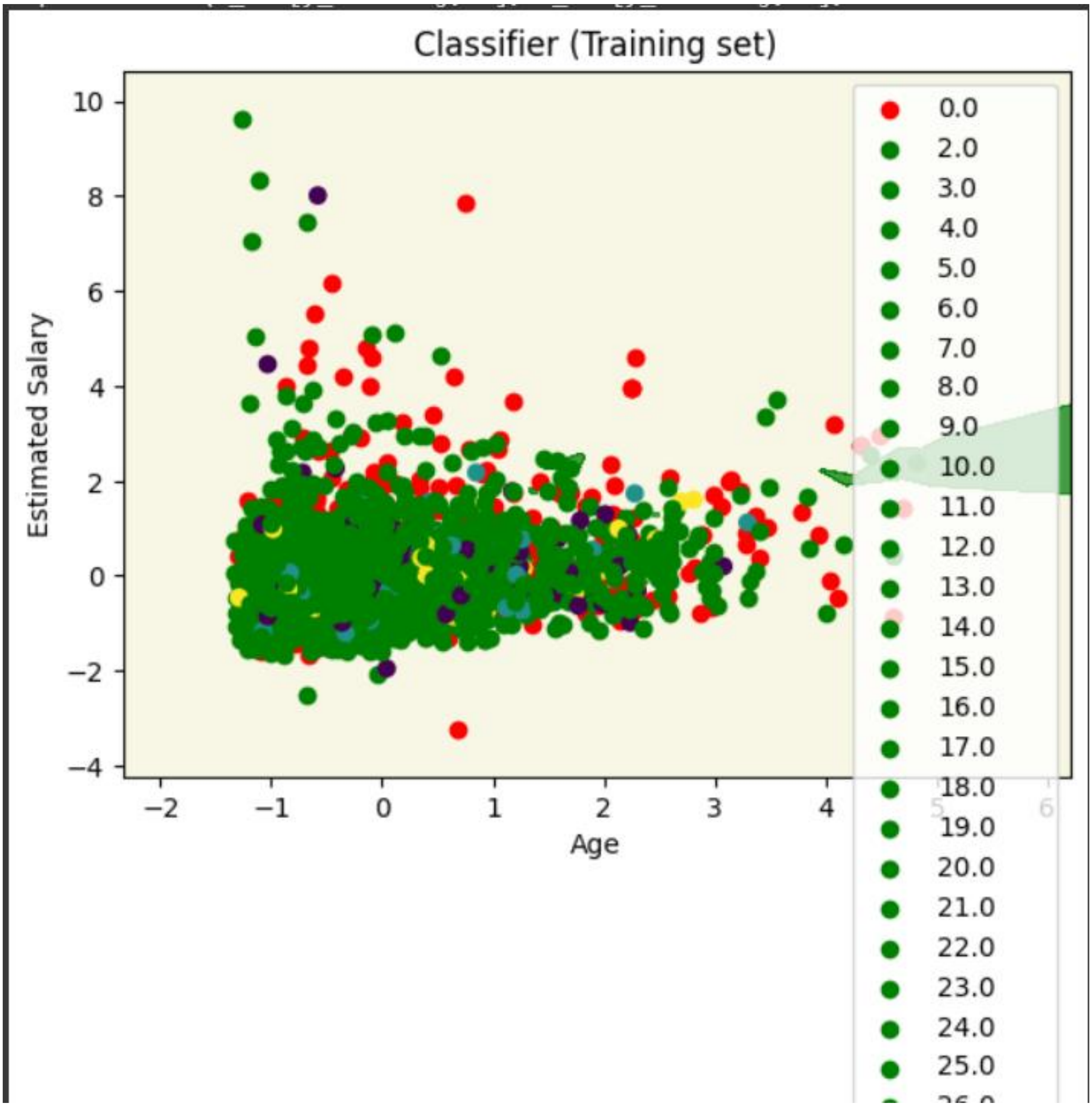


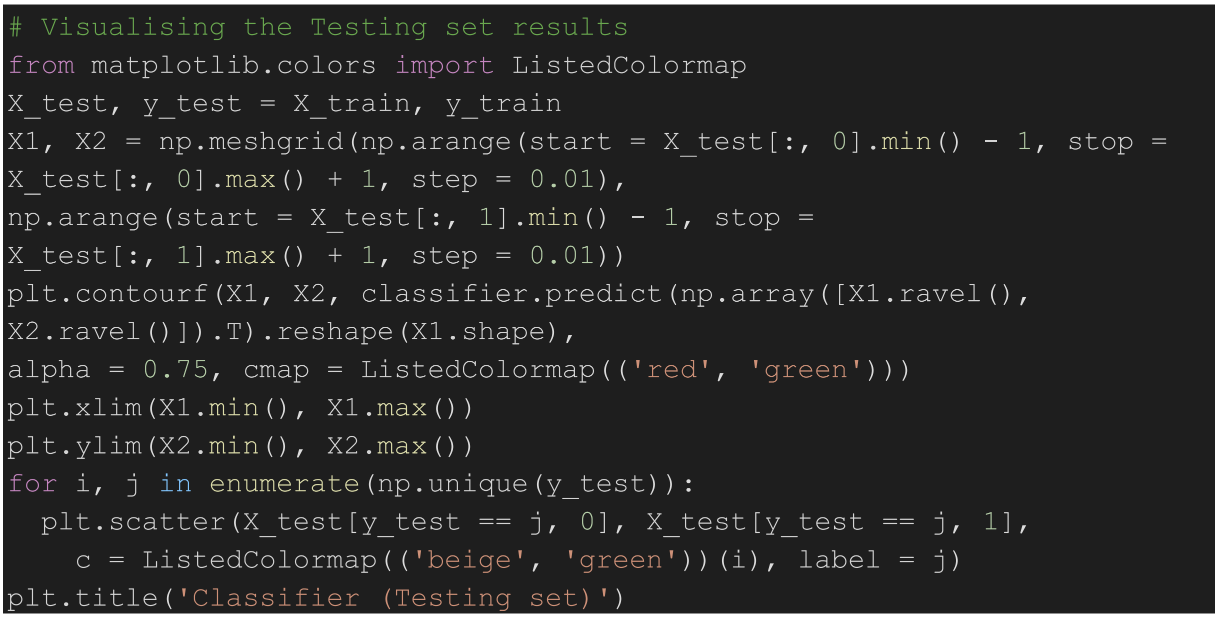


```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('/content/MOVIES DATASET.csv').dropna()
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting classifier to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 2)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

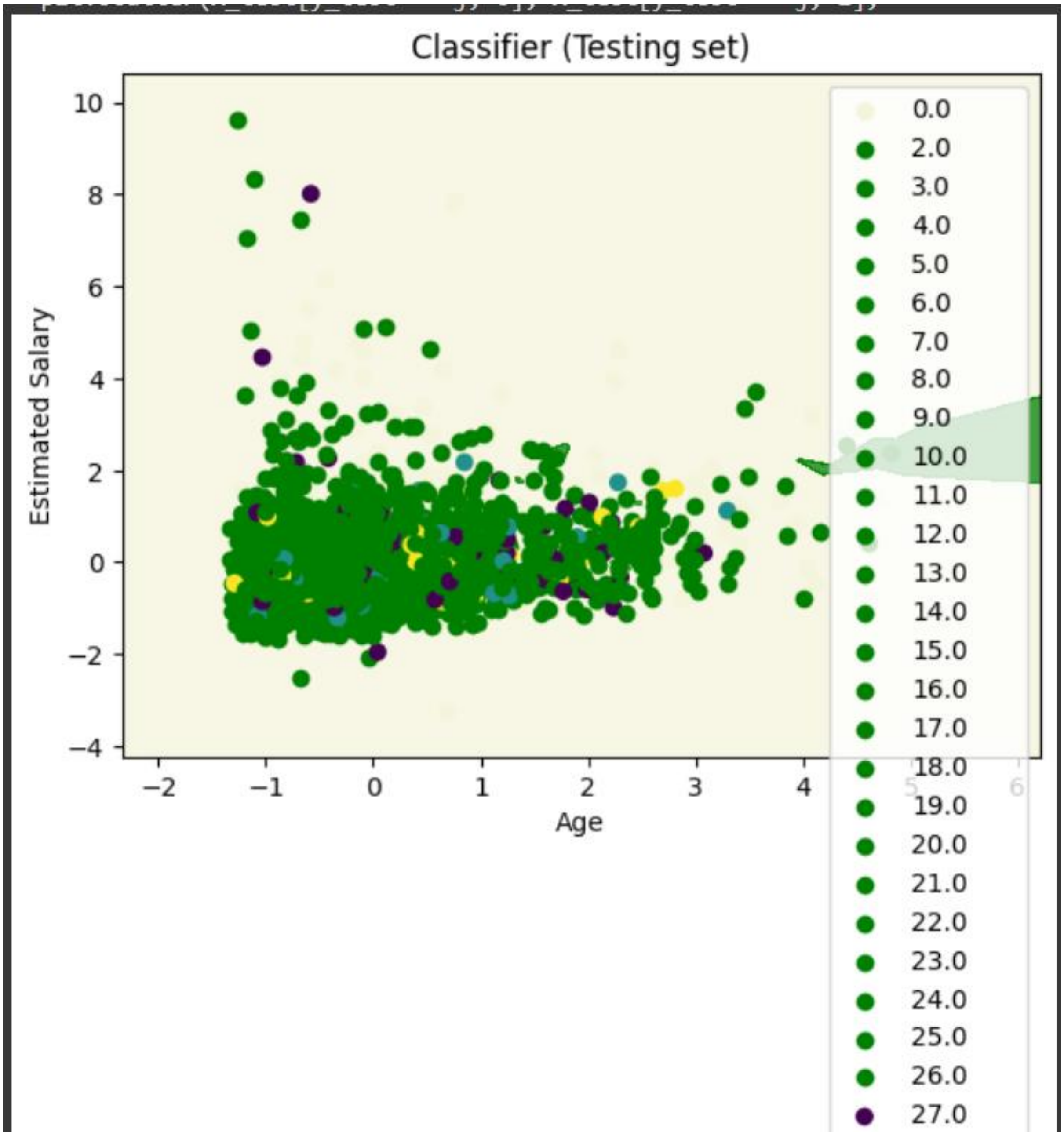


```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('beige', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
    c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Classifier (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```





```
# Visualising the Testing set results
from matplotlib.colors import ListedColormap
X_test, y_test = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_test[:, 0].min() - 1, stop =
X_test[:, 0].max() + 1, step = 0.01),
np.arange(start = X_test[:, 1].min() - 1, stop =
X_test[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_test)):
  plt.scatter(X_test[y_test == j, 0], X_test[y_test == j, 1],
    c = ListedColormap(('beige', 'green'))(i), label = j)
plt.title('Classifier (Testing set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```


Classifier (Testing set)
Estimated Salary
Age