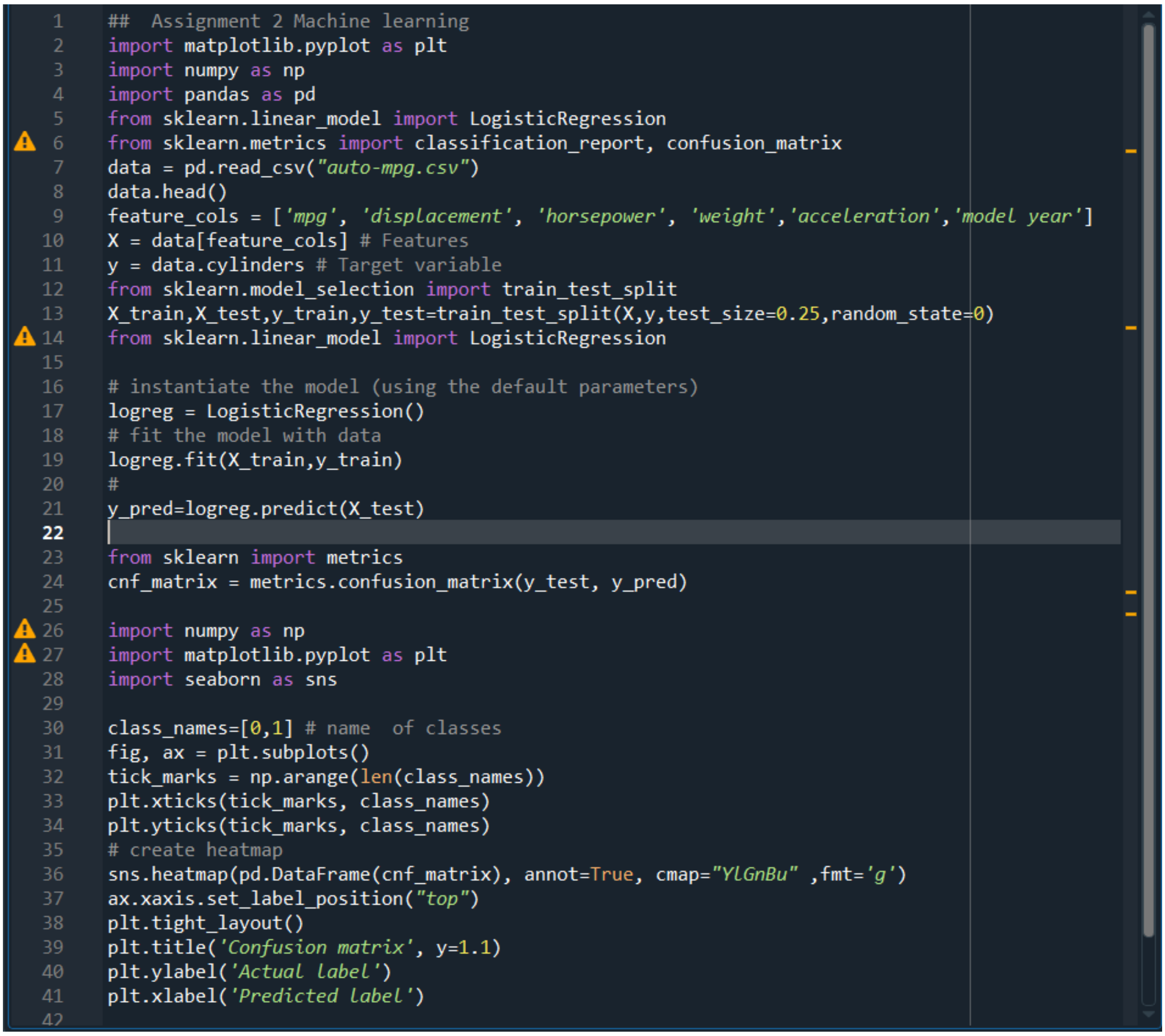
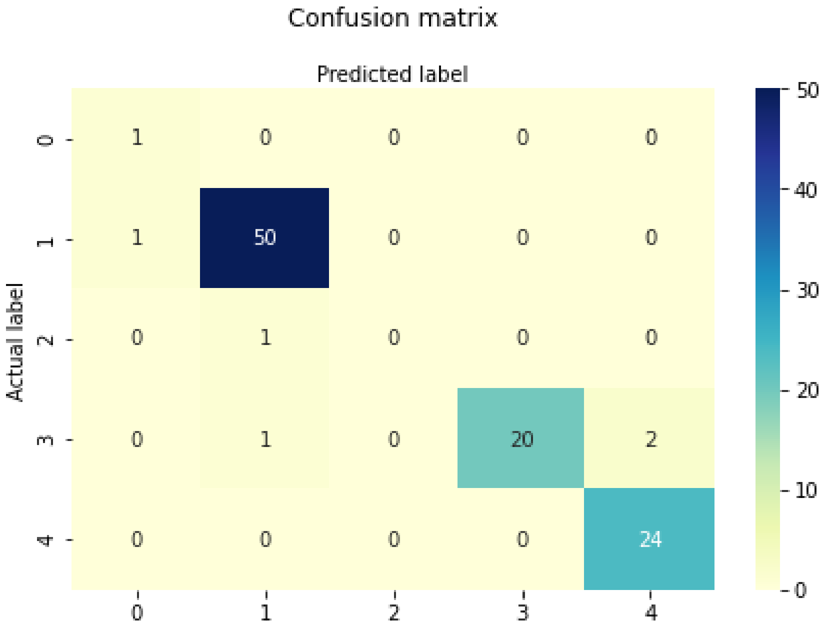
Confusion Matrix:

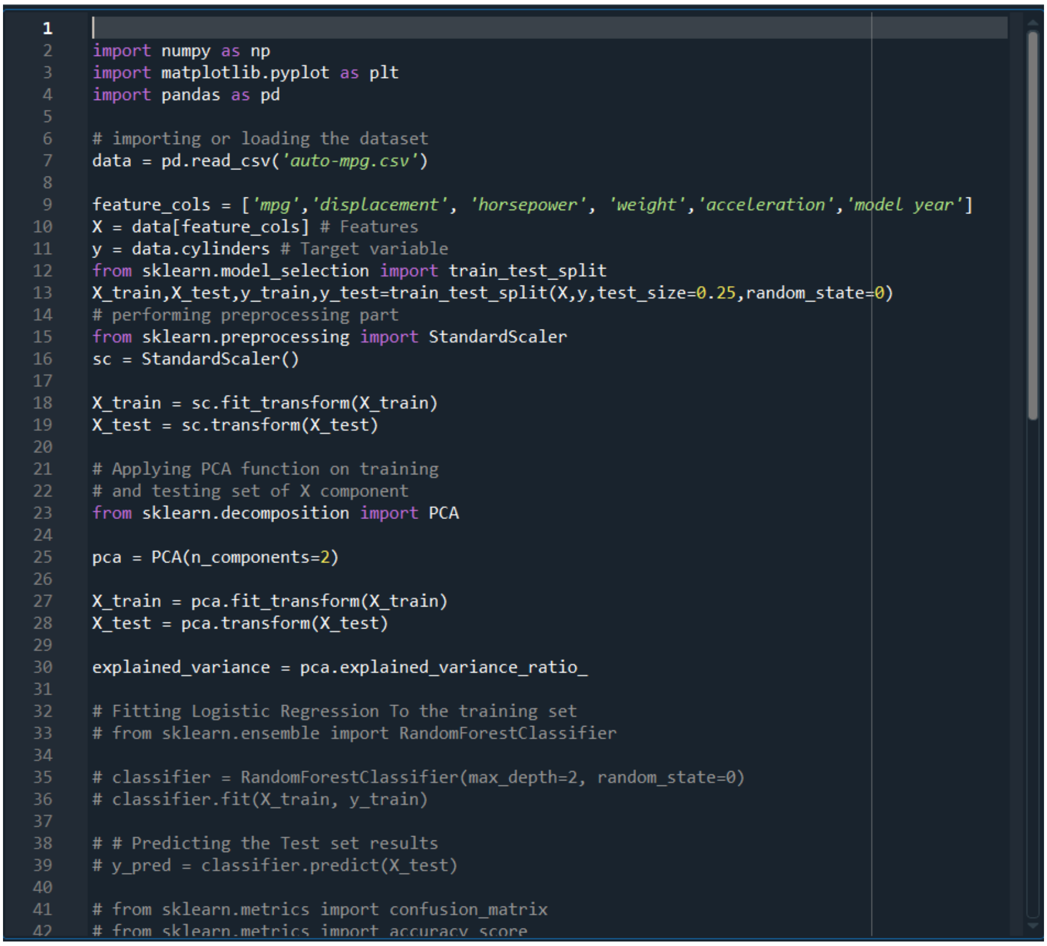


```
##  Assignment 2 Machine learning
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
data = pd.read_csv("auto-mpg.csv")
data.head()
feature_cols = ['mpg', 'displacement', 'horsepower', 'weight','acceleration','model year']
X = data[feature_cols] # Features
y = data.cylinders # Target variable
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.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression()
# fit the model with data
logreg.fit(X_train,y_train)
#
y_pred=logreg.predict(X_test)

from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

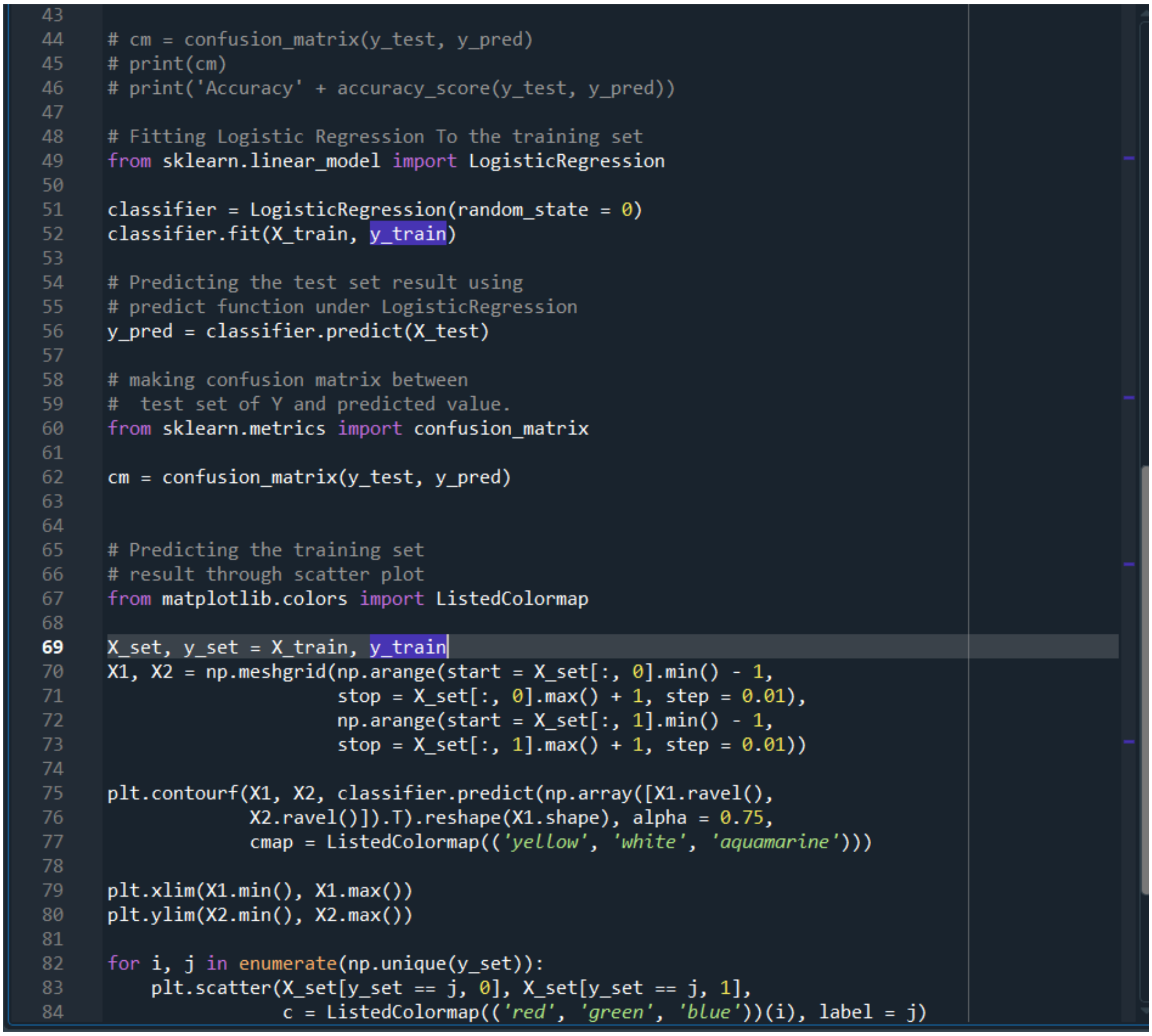
OUTPUT:



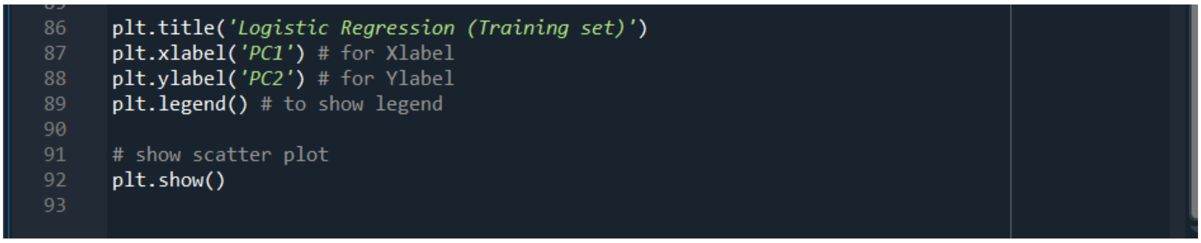
PCA Analysis:



```

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# importing or loading the dataset
data = pd.read_csv('auto-mpg.csv')

feature_cols = ['mpg','displacement', 'horsepower', 'weight','acceleration','model year']
X = data[feature_cols] # Features
y = data.cylinders # Target variable
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)
# performing preprocessing part
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Applying PCA function on training
# and testing set of X component
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_

# Fitting Logistic Regression To the training set
# from sklearn.ensemble import RandomForestClassifier

# classifier = RandomForestClassifier(max_depth=2, random_state=0)
# classifier.fit(X_train, y_train)

# # Predicting the Test set results
# y_pred = classifier.predict(X_test)

# from sklearn.metrics import confusion_matrix
# from sklearn.metrics import accuracy_score
```

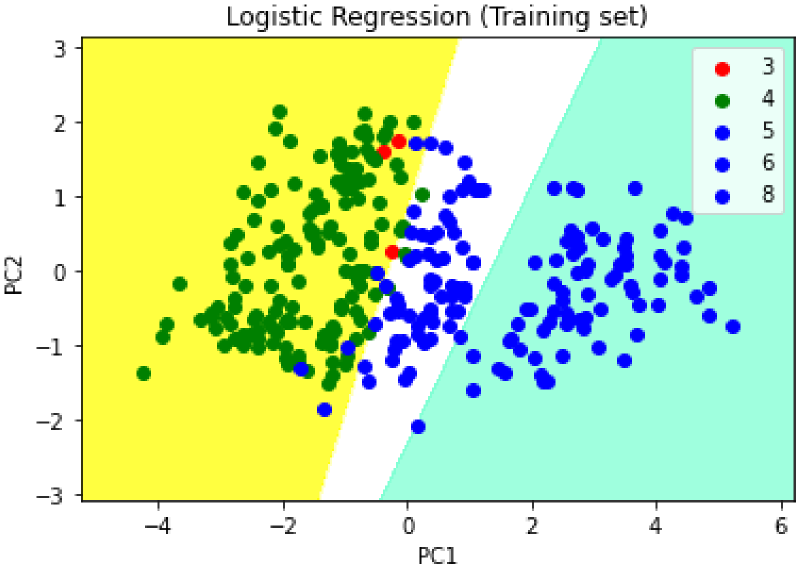


```
# cm = confusion_matrix(y_test, y_pred)
# print(cm)
# print('Accuracy' + accuracy_score(y_test, y_pred))

# Fitting Logistic Regression To the training set
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the test set result using
# predict function under LogisticRegression
y_pred = classifier.predict(X_test)

# making confusion matrix between
#  test set of Y and predicted value.
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)


# Predicting the training set
# result through scatter plot
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(('yellow', 'white', 'aquamarine')))

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', 'blue'))(i), label = j)
```

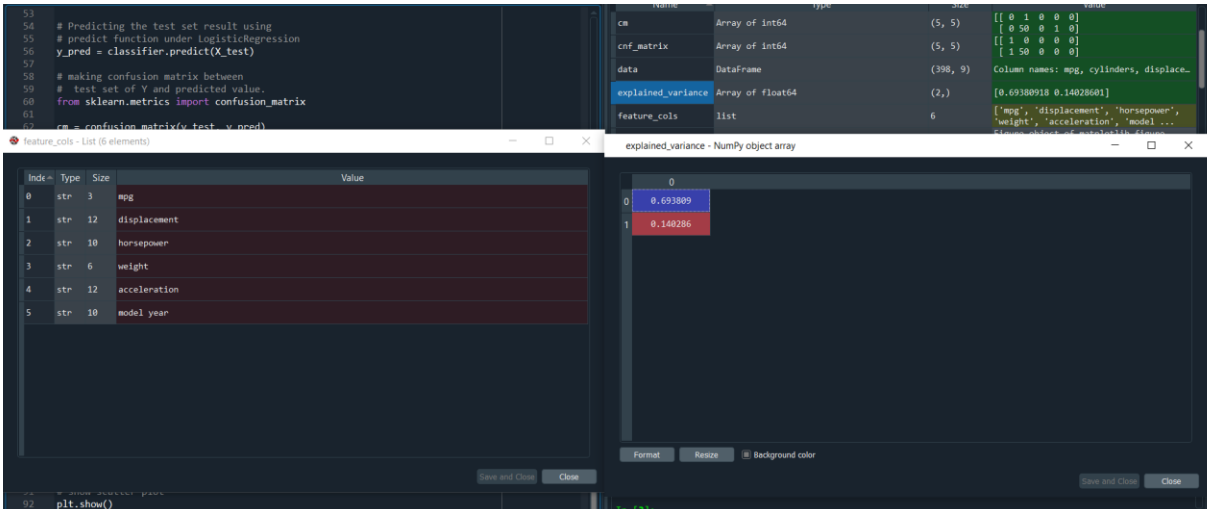


```
plt.title('Logistic Regression (Training set)')
plt.xlabel('PC1') # for Xlabel
plt.ylabel('PC2') # for Ylabel
plt.legend() # to show legend

# show scatter plot
plt.show()
```

OUTPUT:



Variables:



```
# Predicting the test set result using
# predict function under LogisticRegression
y_pred = classifier.predict(X_test)

# making confusion matrix between
#  test set of Y and predicted value.
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
```

Name	Type	Size	Value
cm	Array of int64	(5, 5)	[[0 1 0 0 0] [0 50 0 1 0]
cnf_matrix	Array of int64	(5, 5)	[[1 0 0 0 0] [1 50 0 0 0]
data	DataFrame	(398, 9)	Column names: mpg, cylinders, displace…
explained_variance	Array of float64	(2,)	[0.69380918 0.14028601]
feature_cols	list	6	['mpg', 'displacement', 'horsepower', 'weight', 'acceleration', 'model ...

feature_cols - List (6 elements)

Inde	Type	Size	Value
0	str	3	mpg
1	str	12	displacement
2	str	10	horsepower
3	str	6	weight
4	str	12	acceleration
5	str	10	model year

explained_variance - NumPy object array

	0
0	0.693809
1	0.140286

```
# show scatter plot
plt.show()
```