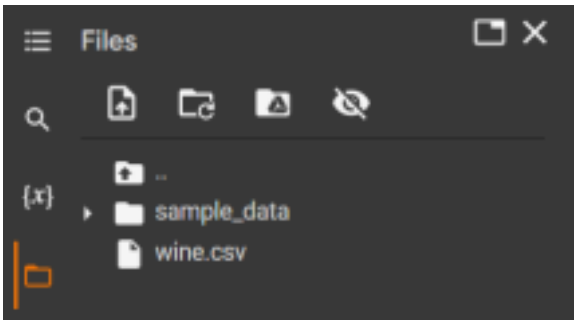
# Practical 4

## Implement PCA

**Program with Output:**

Upload Mall_Customers.csv file




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


# importing or loading the dataset
dataset = pd.read_csv('wine.csv')

# distributing the dataset into two components X and
Y X = dataset.iloc[:, 0:13].values
y = dataset.iloc[:, 13].values
X
```

Kawar Nilesh Ramesh Roll no : 59 Batch A

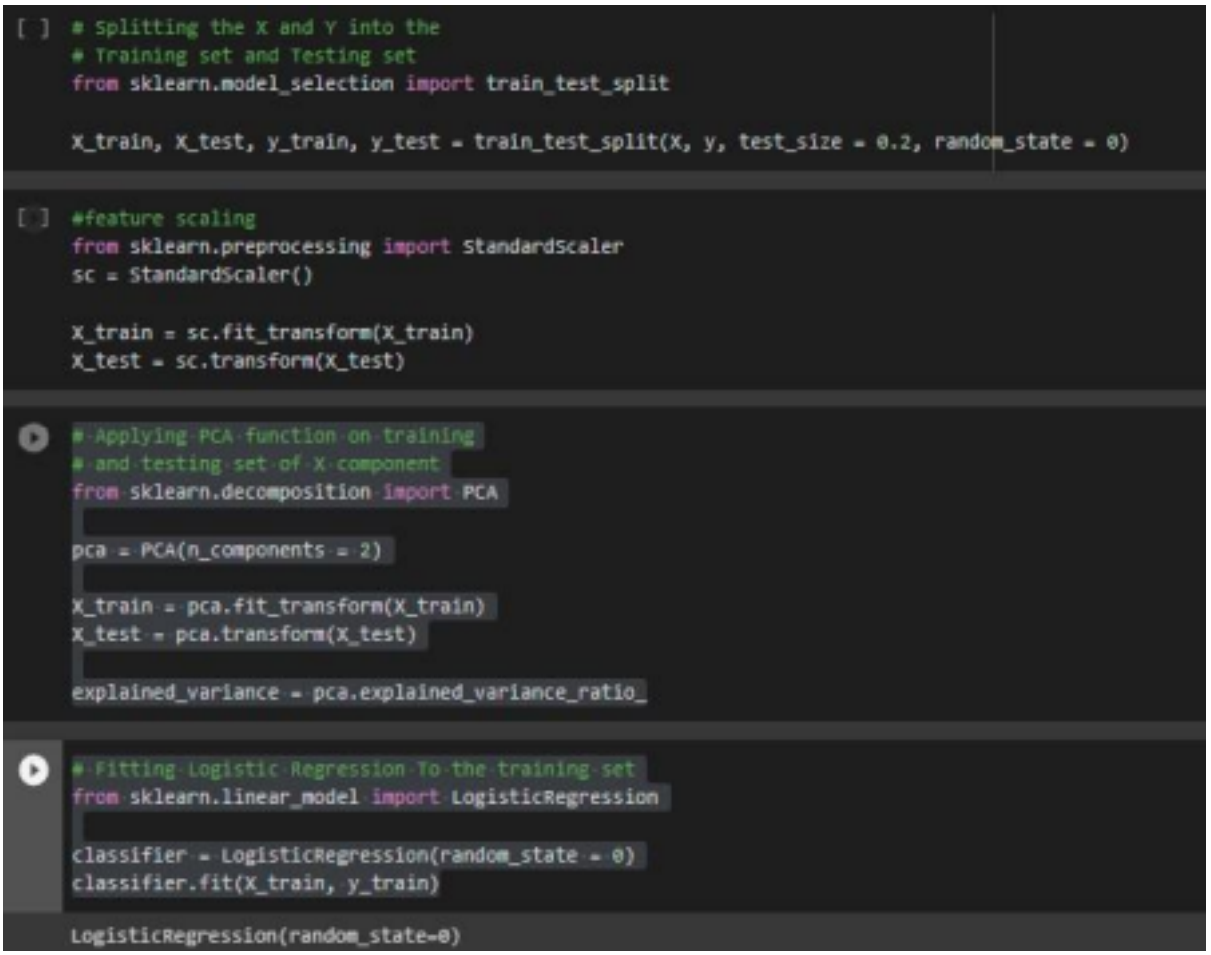
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
# importing or loading the dataset
dataset = pd.read_csv('wine.csv')

# distributing the dataset into two components X and Y
X = dataset.iloc[:, 0:13].values
y = dataset.iloc[:, 13].values
X
```

```
array([[ 1.  , 14.23,  1.71, ...,  5.64,  1.04,  3.92],
       [ 1.  , 13.2 ,  1.78, ...,  4.38,  1.05,  3.4 ],
       [ 1.  , 13.16,  2.36, ...,  5.68,  1.03,  3.17],
       ...,
       [ 3.  , 13.27,  4.28, ..., 10.2 ,  0.59,  1.56],
       [ 3.  , 13.17,  2.59, ...,  9.3 ,  0.6 ,  1.62],
       [ 3.  , 14.13,  4.1 , ...,  9.2 ,  0.61,  1.6 ]])
```

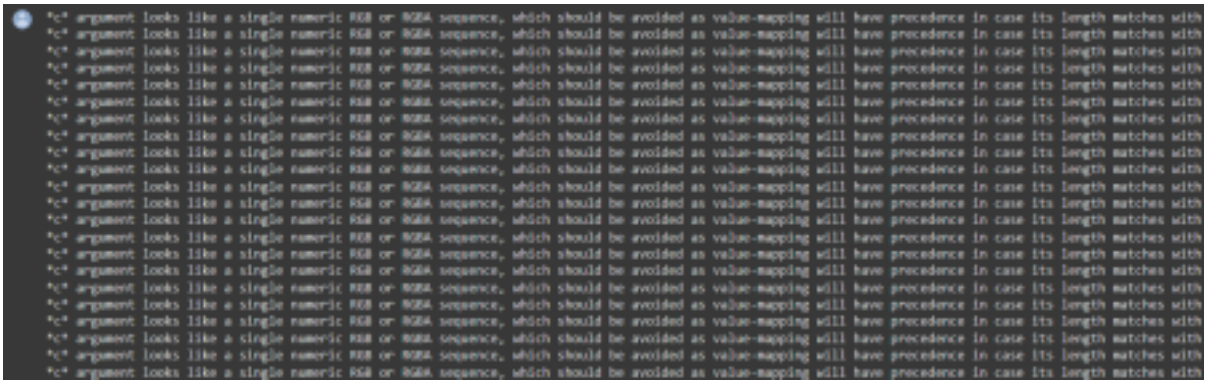
```
# Splitting the X and Y into the
# Training set and Testing set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0 .2, random_state = 0)


#feature scaling
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.linear_model import LogisticRegression

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



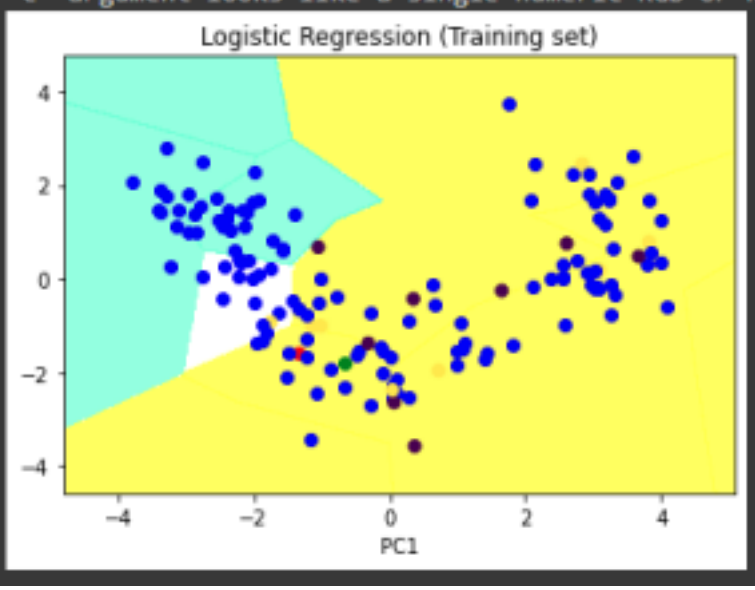

```
# Predicting the test set result using
# predict function under LogisticRegression
y_pred = classifier.predict(X_test)


# 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
```

```
#Create the grid. step=0.01 means all the pixels were actually with
#a 0.01 resolution. min and max of the
#X_Set use with minus ana plus one to prevent ponits to be squeezed
 #on the axes
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


# show scatter plot

plt.show()
```




Kawar Nilesh Ramesh Roll no : 59 Batch A

```
*c* argument looks like a single numeric RGB or RGBA sequence,
*c* argument looks like a single numeric RGB or RGBA sequence,
*c* argument looks like a single numeric RGB or RGBA sequence,
```




```
# Visualising the Test set results through scatter
plot from matplotlib.colors import ListedColormap

X_set, y_set = X_test, y_test

  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)

# title for scatter plot
plt.title('Logistic Regression (Test set)')
plt.xlabel('PC1') # for Xlabel
plt.ylabel('PC2') # for Ylabel
```

Kawar Nilesh Ramesh Roll no : 59 Batch A

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x*
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x*
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x*
```