Principal Component Analysis (PCA)

Importing the libraries

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

Importing the dataset

```
dataset = pd.read_csv('Wine.csv')

dataset.shape

(178, 14)

dataset.head(1)

   Alcohol  Malic_Acid   Ash  Ash_Alcanity  Magnesium
Total_Phenols  \
0    14.23        1.71  2.43          15.6        127            2.8


   Flavanoids  Nonflavanoid_Phenols  Proanthocyanins  Color_Intensity
Hue  \
0        3.06                  0.28             2.29             5.64
1.04

   OD280  Proline  Customer_Segment
0   3.92     1065                 1

X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].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 =
0.2, 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)

X_train.shape
```

```
(142, 13)
```

Applying PCA

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 4)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

```
X_train.shape
```

```
(142, 4)
```

Training the Logistic Regression model on the Training set

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
C:\Users\Dell\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression(random_state=0)
```

Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[13  1  0]
 [ 0 14  2]
 [ 0  0  6]]
```

```
0.9166666666666666
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           1       1.00      0.93      0.96        14
           2       0.93      0.88      0.90        16
           3       0.75      1.00      0.86         6

    accuracy                           0.92        36
   macro avg       0.89      0.93      0.91        36
weighted avg       0.93      0.92      0.92        36
```