```python

# importing required libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```python
# 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
print(X,y)
```

```
[[1.423e+01 1.710e+00 2.430e+00 ... 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 ... 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 ... 1.030e+00 3.170e+00 1.185e+03]
 ...
 [1.327e+01 4.280e+00 2.260e+00 ... 5.900e-01 1.560e+00 8.350e+02]
 [1.317e+01 2.590e+00 2.370e+00 ... 6.000e-01 1.620e+00 8.400e+02]
 [1.413e+01 4.100e+00 2.740e+00 ... 6.100e-01 1.600e+00 5.600e+02]] [1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
```

```python
# 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)
```

```python
# performing preprocessing part
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

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

```python
# 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_
```

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

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

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

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

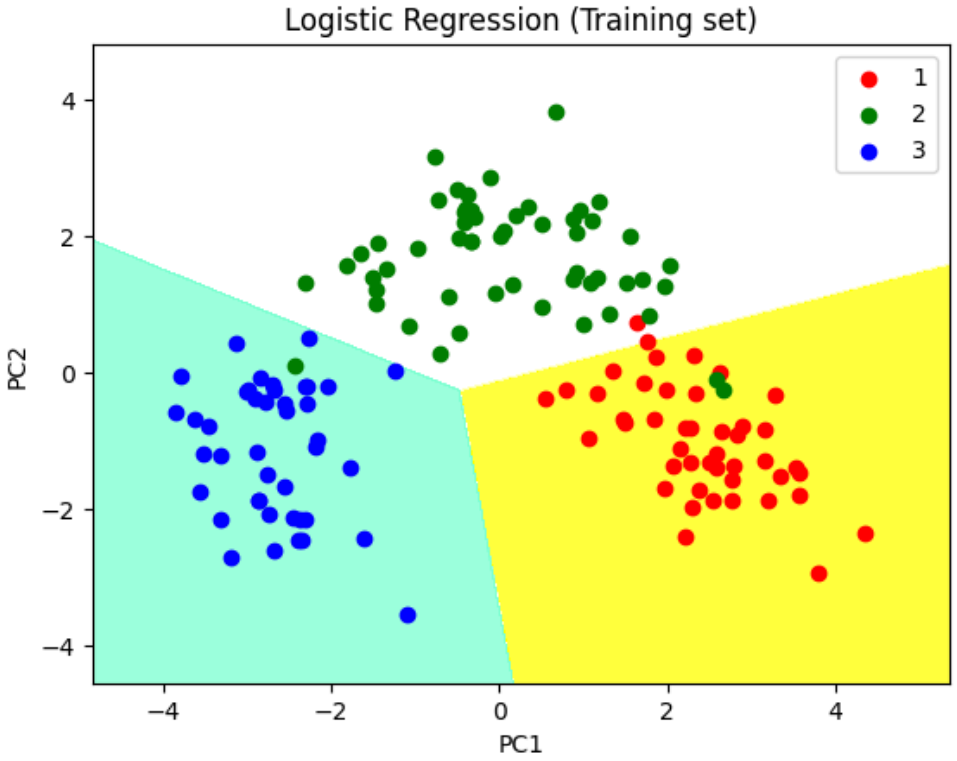
```
array([1, 3, 2, 1, 2, 1, 1, 3, 2, 2, 3, 3, 1, 2, 3, 2, 1, 1, 2, 1, 2, 1,
       1, 2, 2, 2, 2, 2, 2, 3, 1, 1, 2, 1, 1, 1])
```

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

cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[14,  0,  0],
       [ 1, 15,  0],
       [ 0,  0,  6]])
```

```
 # 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()
```

```
<ipython-input-30-450bc7bd07a4>:19: UserWarning: *c* argument looks like a single
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```




```
# 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
plt.legend()

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

```
29   plt.show()
```

```
<ipython-input-31-19421c1f6c28>:19: UserWarning: *c* argument looks like a single
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```