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

```
df = pd.read_csv('wine.csv')
```

```
df.head()
```

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflav
0	14.23	1.71	2.43	15.6	127	2.80	3.06	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	

```
df.tail()
```

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonfl
173	13.71	5.65	2.45	20.5	95	1.68	0.61	
174	13.40	3.91	2.48	23.0	102	1.80	0.75	
175	13.27	4.28	2.26	20.0	120	1.59	0.69	
176	13.17	2.59	2.37	20.0	120	1.65	0.68	
177	14.13	4.10	2.74	24.5	96	2.05	0.76	

```
df.isnull().sum()
```

```
Alcohol                0
Malic_Acid             0
Ash                    0
Ash_Alcanity           0
Magnesium              0
Total_Phenols          0
Flavanoids             0
Nonflavanoid_Phenols   0
Proanthocyanins        0
Color_Intensity        0
Hue                    0
OD280                  0
Proline                0
Customer_Segment       0
dtype: int64
```

```
df.describe()
```

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Colo
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.590899	
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Alcohol               178 non-null    float64
 1   Malic_Acid            178 non-null    float64
 2   Ash                   178 non-null    float64
 3   Ash_Alcanity          178 non-null    float64
 4   Magnesium             178 non-null    int64  
 5   Total_Phenols         178 non-null    float64
 6   Flavanoids            178 non-null    float64
 7   Nonflavanoid_Phenols  178 non-null    float64
 8   Proanthocyanins       178 non-null    float64
 9   Color_Intensity       178 non-null    float64
 10  Hue                   178 non-null    float64
 11  OD280                 178 non-null    float64
 12  Proline               178 non-null    int64  
 13  Customer_Segment      178 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

```
df.dropna(inplace=True)
```

```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_ = sc.fit_transform(X)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, ytrain, ytest = train_test_split(X_,y, test_size = 0.2, random_state = 0)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.fit_transform(X_test)
```

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

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

```
from sklearn.metrics import confusion_matrix, accuracy_score
ypred = classifier.predict(X_test)
cm = confusion_matrix(ytest,ypred)
ac = accuracy_score(ytest,ypred)
print("Confusion Matrix: \n", cm)
print()
print("Accuracy Score: \n", ac)
```

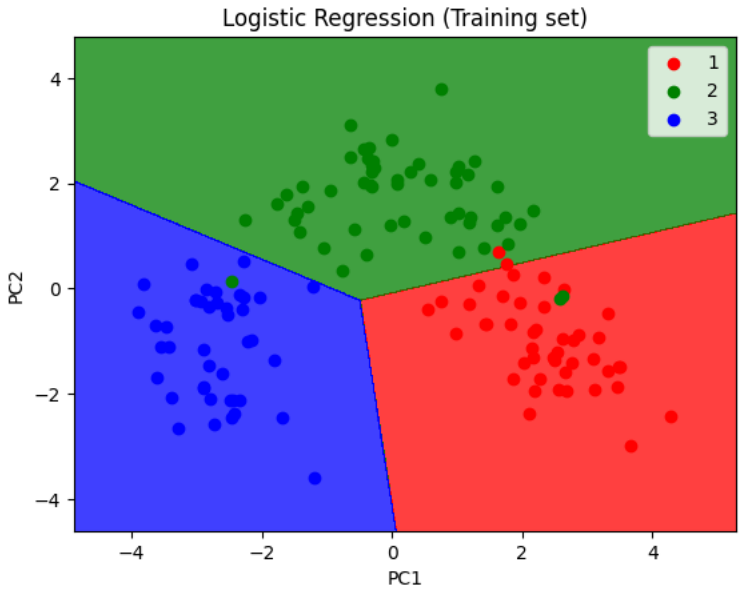
```
Confusion Matrix:
 [[ 0  1 13]
 [ 4 11  1]
 [ 6  0  0]]

Accuracy Score:
 0.3055555555555556
```

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, ytrain
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(('red', 'green', 'blue')))
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')
plt.ylabel('PC2')
plt.legend()
plt.show()
```

```
<ipython-input-20-a276a67f5d16>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided a
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
```




```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, ytest
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(('red', 'green', 'blue')))
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 (Test set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()
```

```
<ipython-input-21-f8018a0b205a>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided a
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
```