UCS2612 Machine Learning Laboratory

A9. Applications of dimensionality reductiontechniques

Name : Mega V
Roll No : 3122 21 5001 051

Aim

Develop a python program to perform dimensionality reduction using PCA and LDA. Visualize the features from the dataset and interpret the results obtained by the model using Matplotlib library.

Dataset:- http://www3.dsi.uminho.pt/pcortez/wine/winequality.zip

Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
```

Read dataset

```
# importing or loading the dataset
data = pd.read_csv(""C:\Users\SSN\Desktop\ML
Lab\A9\winequality.zip"",header=0, sep=";")

data.head()
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0            7.4              0.70         0.00             1.9
0.076
1            7.8              0.88         0.00             2.6
0.098
2            7.8              0.76         0.04             2.3
0.092
3           11.2              0.28         0.56             1.9
0.075
4            7.4              0.70         0.00             1.9
0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
0                 11.0                  34.0   0.9978  3.51       0.56

1                 25.0                  67.0   0.9968  3.20       0.68
```

```
2                 15.0                  54.0   0.9970  3.26       0.65

3                 17.0                  60.0   0.9980  3.16       0.58

4                 11.0                  34.0   0.9978  3.51       0.56


   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5

data.describe()

       fixed acidity  volatile acidity   citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.390000     0.090000        1.900000
50%         7.900000          0.520000     0.260000        2.200000
75%         9.200000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide
density  \
count  1599.000000           1599.000000           1599.000000
1599.000000
mean      0.087467             15.874922             46.467792
0.996747
std       0.047065             10.460157             32.895324
0.001887
min       0.012000              1.000000              6.000000
0.990070
25%       0.070000              7.000000             22.000000
0.995600
50%       0.079000             14.000000             38.000000
0.996750
75%       0.090000             21.000000             62.000000
0.997835
max       0.611000             72.000000            289.000000
1.003690

                pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
```

```
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
75%       3.400000     0.730000    11.100000     6.000000
max       4.010000     2.000000    14.900000     8.000000
```

Data Pre-processing

```
data.corr()
```

```
                      fixed acidity volatile acidity   citric acid  \
fixed acidity              1.000000         -0.256131     0.671703
volatile acidity          -0.256131          1.000000    -0.552496
citric acid                0.671703         -0.552496     1.000000
residual sugar             0.114777          0.001918     0.143577
chlorides                  0.093705          0.061298     0.203823
free sulfur dioxide       -0.153794         -0.010504    -0.060978
total sulfur dioxide      -0.113181          0.076470     0.035533
density                    0.668047          0.022026     0.364947
pH                        -0.682978          0.234937    -0.541904
sulphates                  0.183006         -0.260987     0.312770
alcohol                   -0.061668         -0.202288     0.109903
quality                    0.124052         -0.390558     0.226373

                      residual sugar  chlorides  free sulfur
dioxide  \
fixed acidity               0.114777   0.093705            -0.153794

volatile acidity            0.001918   0.061298            -0.010504

citric acid                 0.143577   0.203823            -0.060978

residual sugar              1.000000   0.055610             0.187049

chlorides                   0.055610   1.000000             0.005562

free sulfur dioxide         0.187049   0.005562             1.000000

total sulfur dioxide        0.203028   0.047400             0.667666

density                     0.355283   0.200632            -0.021946

pH                         -0.085652  -0.265026             0.070377

sulphates                   0.005527   0.371260             0.051658

alcohol                     0.042075  -0.221141            -0.069408

quality                     0.013732  -0.128907            -0.050656
```

```
                      total sulfur dioxide   density        pH
sulphates  \
fixed acidity                    -0.113181  0.668047 -0.682978
0.183006
volatile acidity                  0.076470  0.022026  0.234937  -
0.260987
citric acid                       0.035533  0.364947 -0.541904
0.312770
residual sugar                    0.203028  0.355283 -0.085652
0.005527
chlorides                         0.047400  0.200632 -0.265026
0.371260
free sulfur dioxide               0.667666 -0.021946  0.070377
0.051658
total sulfur dioxide              1.000000  0.071269 -0.066495
0.042947
density                           0.071269  1.000000 -0.341699
0.148506
pH                               -0.066495 -0.341699  1.000000  -
0.196648
sulphates                         0.042947  0.148506 -0.196648
1.000000
alcohol                          -0.205654 -0.496180  0.205633
0.093595
quality                          -0.185100 -0.174919 -0.057731
0.251397

                        alcohol   quality
fixed acidity         -0.061668  0.124052
volatile acidity      -0.202288 -0.390558
citric acid            0.109903  0.226373
residual sugar         0.042075  0.013732
chlorides             -0.221141 -0.128907
free sulfur dioxide   -0.069408 -0.050656
total sulfur dioxide  -0.205654 -0.185100
density               -0.496180 -0.174919
pH                     0.205633 -0.057731
sulphates              0.093595  0.251397
alcohol                1.000000  0.476166
quality                0.476166  1.000000

data.dropna(inplace=True)

scaler_standard = StandardScaler()
data_standardized = scaler_standard.fit_transform(data)

scaler_normal = MinMaxScaler()
data_normalized = scaler_normal.fit_transform(data)
```

```
data_standardized = pd.DataFrame(data_standardized,
columns=data.columns)
data_normalized = pd.DataFrame(data_normalized, columns=data.columns)

data_standardized.head()
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0      -0.528360          0.961877    -1.391472       -0.453218  -
0.243707
1      -0.298547          1.967442    -1.391472        0.043416
0.223875
2      -0.298547          1.297065    -1.186070       -0.169427
0.096353
3       1.654856         -1.384443     1.484154       -0.453218  -
0.264960
4      -0.528360          0.961877    -1.391472       -0.453218  -
0.243707

   free sulfur dioxide  total sulfur dioxide    density        pH
sulphates  \
0            -0.466193             -0.379133  0.558274  1.288643  -
0.579207
1             0.872638              0.624363  0.028261 -0.719933
0.128950
2            -0.083669              0.229047  0.134264 -0.331177  -
0.048089
3             0.107592              0.411500  0.664277 -0.979104  -
0.461180
4            -0.466193             -0.379133  0.558274  1.288643  -
0.579207

    alcohol   quality
0 -0.960246 -0.787823
1 -0.584777 -0.787823
2 -0.584777 -0.787823
3 -0.584777  0.450848
4 -0.960246 -0.787823
```

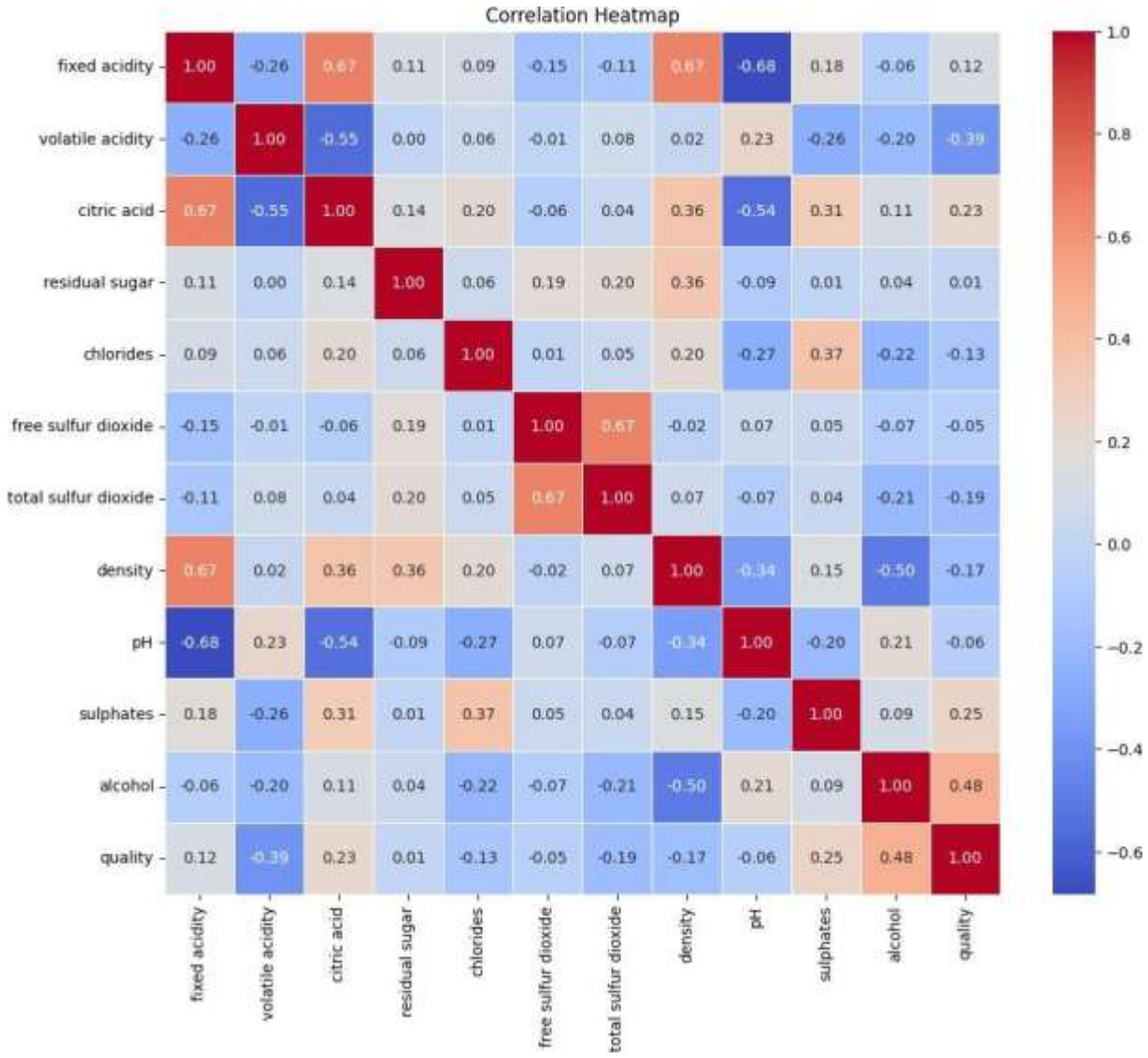
```
data_normalized.head()
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0       0.247788          0.397260         0.00        0.068493
0.106845
1       0.283186          0.520548         0.00        0.116438
0.143573
2       0.283186          0.438356         0.04        0.095890
0.133556
3       0.584071          0.109589         0.56        0.068493
```

```
0.105175
4       0.247788          0.397260         0.00         0.068493
0.106845

   free sulfur dioxide  total sulfur dioxide    density        pH
sulphates  \
0             0.140845              0.098940   0.567548   0.606299
0.137725
1             0.338028              0.215548   0.494126   0.362205
0.209581
2             0.197183              0.169611   0.508811   0.409449
0.191617
3             0.225352              0.190813   0.582232   0.330709
0.149701
4             0.140845              0.098940   0.567548   0.606299
0.137725

    alcohol  quality
0  0.153846      0.4
1  0.215385      0.4
2  0.215385      0.4
3  0.215385      0.6
4  0.153846      0.4
```

EDA

```
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt='.2f',
linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



Splitting the data into testing and training

```
# distributing the dataset into two components X and Y
X_red = data.iloc[:, 0:11].values
y_red = data.iloc[:, 11].values

X_train_red, X_test_red, y_train_red, y_test_red =
train_test_split(X_red,y_red, test_size=0.2, random_state=0)
```

Feature engineering

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

X_train_red = sc.fit_transform(X_train_red)
X_test_red = sc.transform(X_test_red)
```

Building PCA model

```
from sklearn.decomposition import PCA

PCa = PCA(n_components = 2)

X_train_red = PCa.fit_transform(X_train_red)
X_test_red = PCa.transform(X_test_red)

explained_variance = PCa.explained_variance_ratio_

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train_red, y_train_red)
```

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

```
y_pred_red = classifier.predict(X_test_red)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test_red, y_pred_red)
print(cm)
accuracy_score(y_test_red,y_pred_red)
```

```
[[ 0  0  0  2  0  0]
 [ 0  0  4  7  0  0]
 [ 0  0 89 45  1  0]
 [ 0  0 55 81  6  0]
 [ 0  0  4 21  2  0]
 [ 0  0  0  2  1  0]]
```

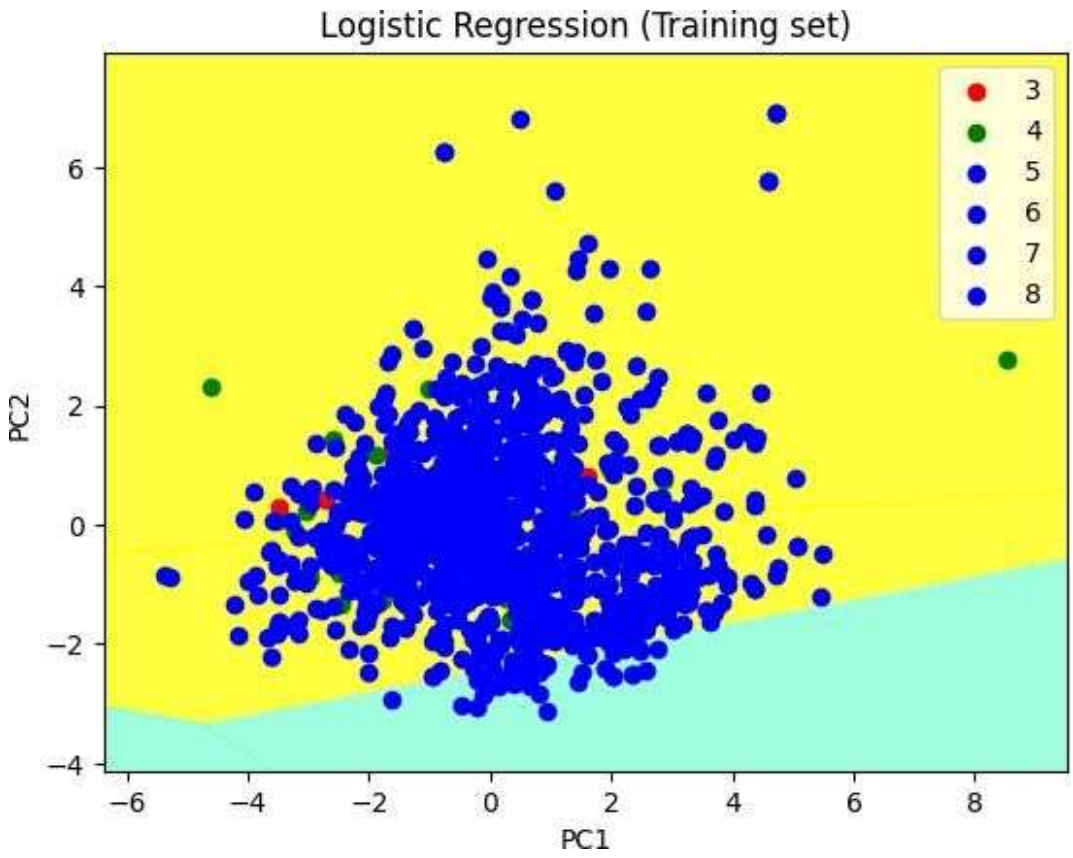
```
0.5375
```

Visualisation of PCA model

```
# result through scatter plot
from matplotlib.colors import ListedColormap
```

```
X_set, y_set = X_train_red, y_train_red
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()
```

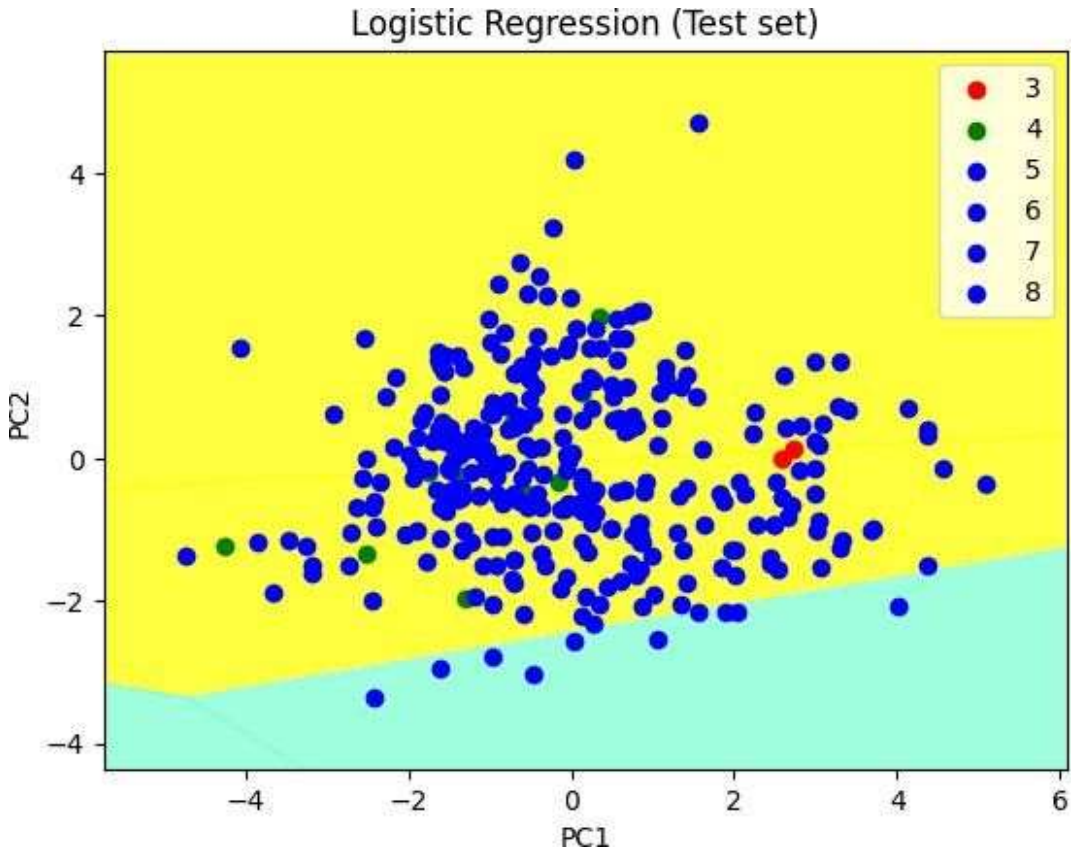
```
C:\Users\nithi\AppData\Local\Temp\ipykernel_4776\4008791166.py:18:
UserWarning: *c* argument looks like a single numeric RGB or RGBA
sequence, which should be avoided as value-mapping will have
precedence in case its length matches with *x* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if
you intend to specify the same RGB or RGBA value for all points.
  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_red, y_test_red

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()
```

```
C:\Users\nithi\AppData\Local\Temp\ipykernel_4776\3957206111.py:19:
UserWarning: *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* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if
you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```



Building LDA model

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
as LDA
```

```
lda = LDA(n_components = 2)
X_train_red = lda.fit_transform(X_train_red, y_train_red)
X_test_red = lda.transform(X_test_red)

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train_red, y_train_red)
```

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

```
y_pred = classifier.predict(X_test_red)
print(y_pred)
```

```
[5 5 6 6 7 5 6 5 6 5 6 6 6 5 5 5 7 6 6 5 6 6 6 5 5 5 5 6 5 6 6 5 5 5 5
6 5
 5 6 6 5 6 6 7 6 5 6 5 6 6 6 5 5 6 5 5 5 6 6 6 5 5 5 6 5 6 6 6 6 6 5 5
5 5
 5 6 6 5 5 6 5 5 6 6 5 5 5 6 6 5 5 5 6 6 6 5 6 6 6 6 6 6 5 6 6 6 6 5 6
5 6
 5 6 5 6 5 6 6 6 6 5 6 5 5 5 6 5 5 6 6 5 6 6 6 6 5 6 5 6 6 6 5 6 5 6 6
6 7
 6 6 6 6 5 6 6 5 6 6 6 5 6 6 6 5 6 5 5 6 5 6 5 5 5 7 5 6 6 6 6 5 5 6 5
5 7
 5 5 5 5 5 6 6 6 6 5 5 6 5 5 5 5 6 5 6 5 5 5 6 6 5 6 5 5 5 5 5 6 6 5 5
5 6
 6 6 5 5 6 6 6 6 5 6 5 5 6 5 6 6 7 6 5 5 5 5 6 5 5 5 5 6 5 5 5 6 6 5 5
6 5
 5 5 6 5 6 6 6 5 5 6 5 6 6 6 6 6 5 6 5 7 6 5 6 6 6 6 5 6 6 6 5 5 6 6 6
5 6
 5 5 5 5 5 6 6 5 6 5 5 5 5 5 5 5 5 6 5 7 6 6 6 7]
```

```
accuracy = accuracy_score(y_test_red, y_pred)
print("Accuracy of LDA model:", accuracy)
```

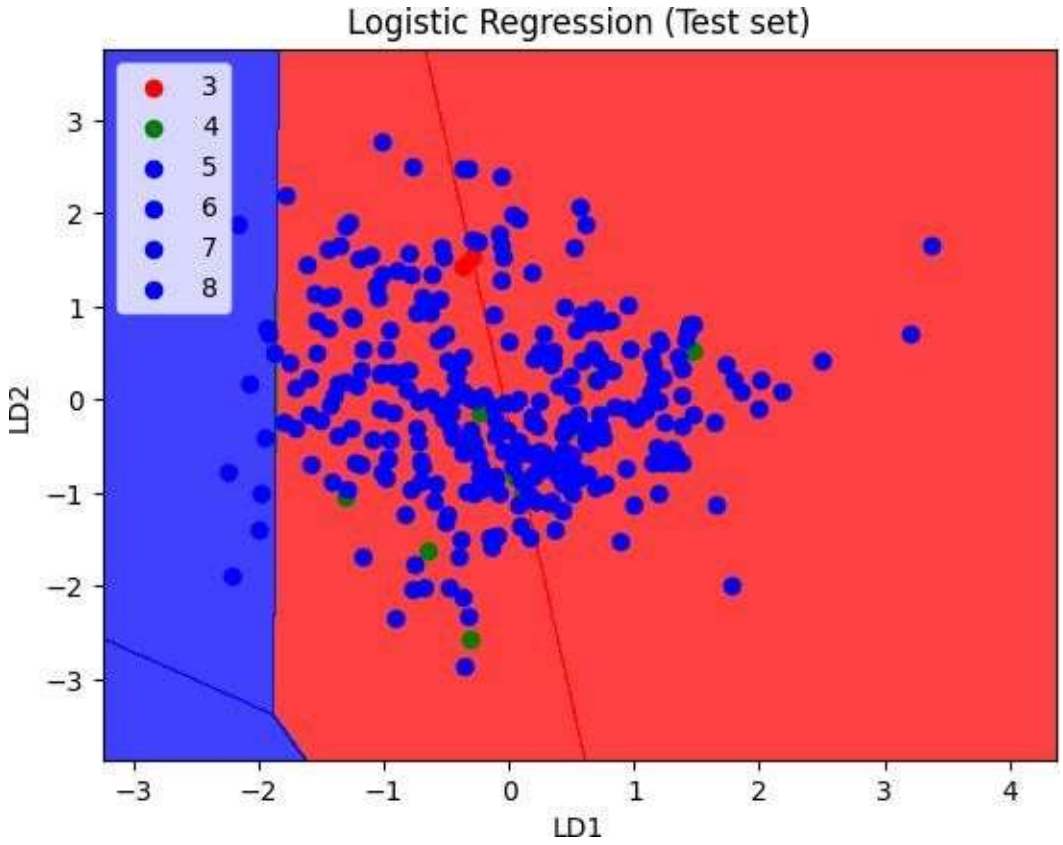
```
Accuracy of LDA model: 0.5375
```

Visualisation of LDA model

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test_red, y_test_red

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')))
```

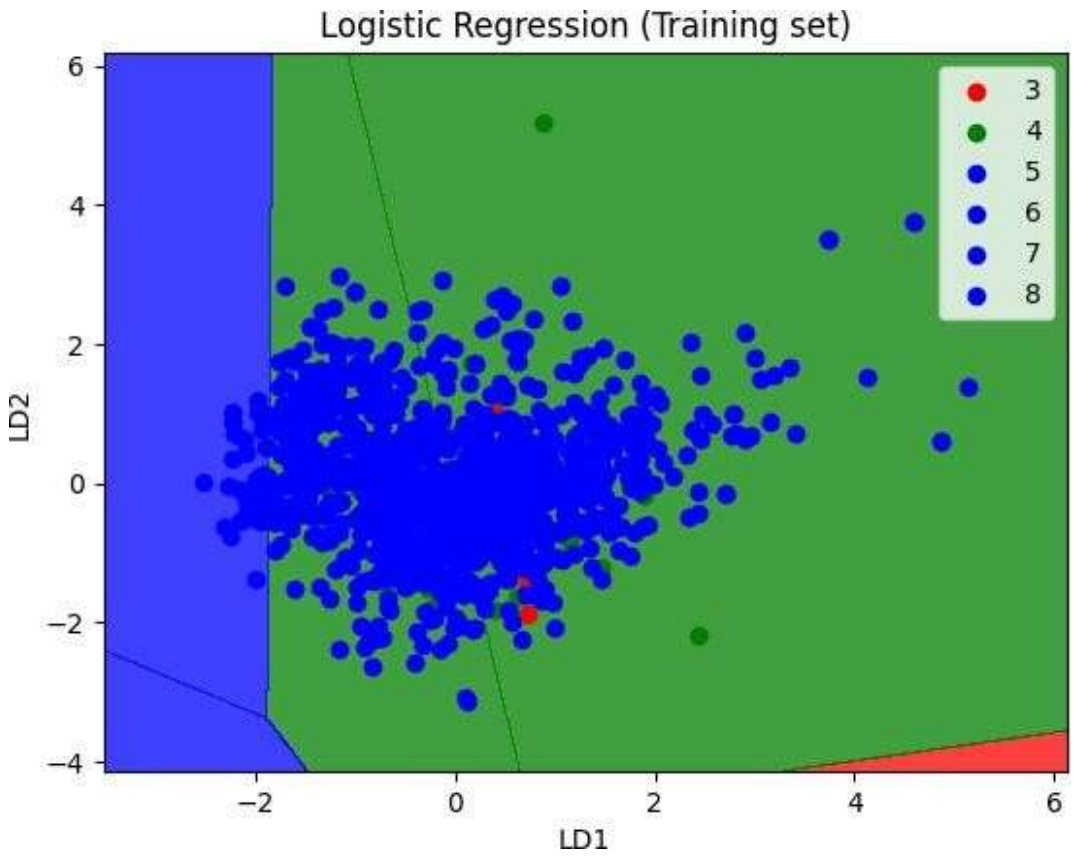
```
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

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('LD1')
plt.ylabel('LD2')
plt.legend()
plt.show()
```

```
C:\Users\nithi\AppData\Local\Temp\ipykernel_4776\531035640.py:12:
UserWarning: *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* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if
you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```



```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train_red, y_train_red

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('LD1')
plt.ylabel('LD2')
plt.legend()
plt.show()
```

```
C:\Users\nithi\AppData\Local\Temp\ipykernel_4776\171321940.py:12:
UserWarning: *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* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if
you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```



Inference

1. After applying PCA, you can analyze the principal components to understand which original features contribute the most to the variance in the data. You can also visualize the data in reduced dimensions to explore patterns or clusters.

2. After applying LDA, you can interpret the learned linear discriminants to understand how the classes are separated in the reduced-dimensional space. LDA provides insight into which features are most discriminative for class separation.

Learning Outcomes

1. Implementation of Pre-processing, EDA and feature selection.
2. Implementation of PCA nad LDA models and visualising it.
3. Displaying the confusion matrix.
4. Understanding the techniques of dimentionality reduction.