PRACTICAL 7

AIM: Understand feature selection and feature extraction techniques in detail.
Implement feature extraction techniques (PCA and LDA) on the given dataset, wine.csv

Read Dataset

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

```
data =
pd.read_csv("/content/drive/MyDrive/ML_Collage/Wine .csv
")
```

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	Hue	OD280	Proline	Customer_Segment
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065	1
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050	1
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185	1
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480	1
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740	3
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750	3
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835	3
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840	3
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560	3

```
data.isnull().any(axis=1).sum()
```

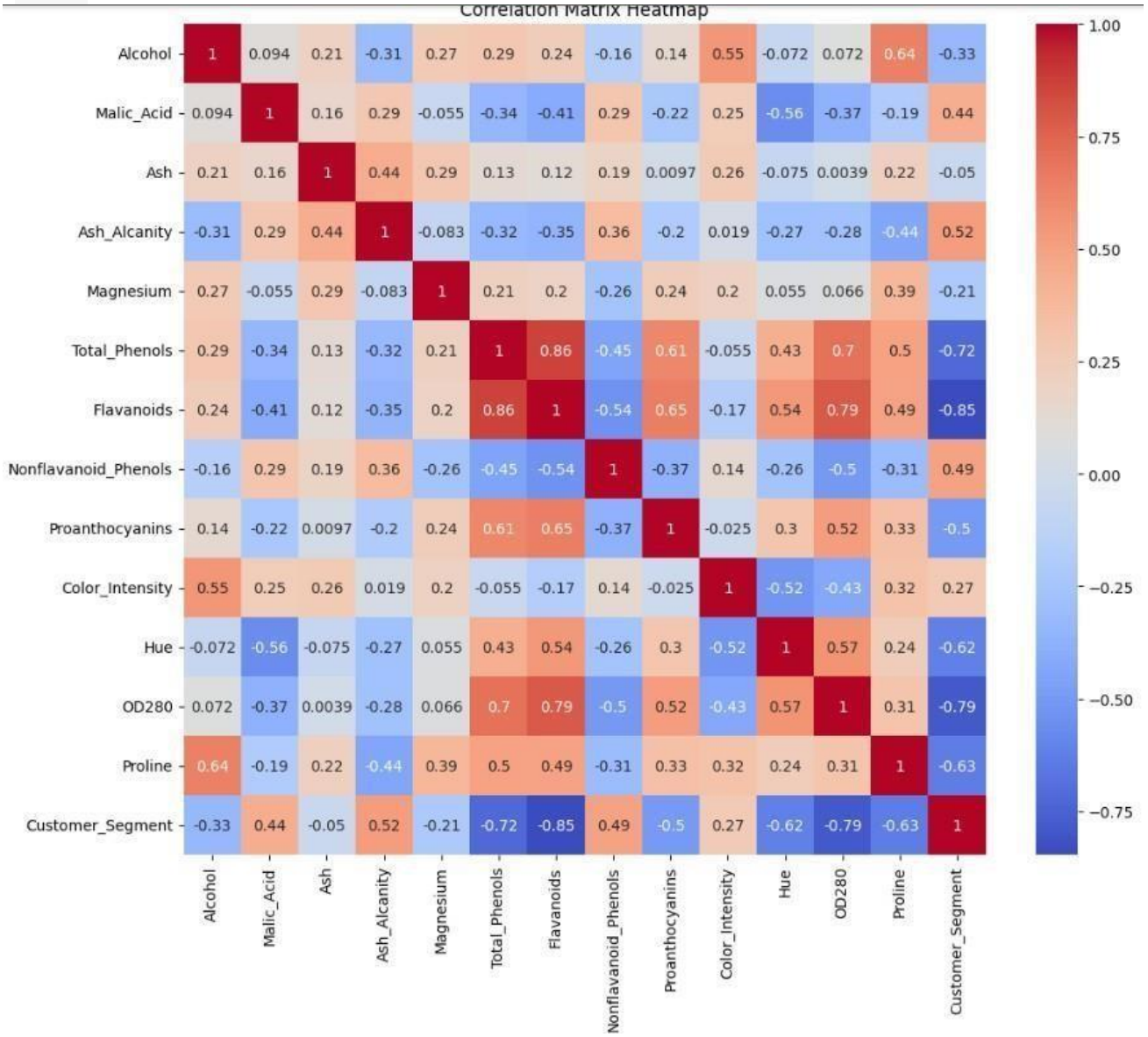
0

```
data["Customer_Segment"].value_counts()
```

Customer_Segment	
2	71
1	59
3	48

Name: DHRUV SHERE
Enrollment No: 23012022021
Batch: 4IT-B-2

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix,
annot=True, cmap='coolwarm')
plt.title('Correlation Matrix
Heatmap')
plt.show()
```



DataSplitting

```
from sklearn.model_selection import
train_test_split
```

```
x = data.drop("Customer_Segment", axis=1) y
= data["Customer_Segment"]

x_train, x_test, y_train, y_test =
train_test_split(x, y, test_size=0.3,
random_state=42)
```

```
x_train.shape, x_test.shape, y_train.shape,
y_test.shape
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler() x_train =
scaler.fit_transform(x_train) x_test =
scaler.transform(x_test)
```

PCA

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

```
lg = LogisticRegression()
lg.fit(X_train, y_train)
y_pred = lg.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print('--------------------------------------------------')
print(classification_report(y_test, y_pred))
print('--------------------------------------------------')
print(accuracy_score(y_test, y_pred))
```

```
[[13  2  0]
 [ 0 18  0]
 [ 0  0 12]]
---------------------------------
              precision    recall  f1-score   support

           1       1.00      0.87      0.93        15
           2       0.90      1.00      0.95        18
           3       1.00      1.00      1.00        12

    accuracy                           0.96        45
   macro avg       0.97      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45

---------------------------------
0.9555555555555556
```

LDA

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression() model.fit(x_train,
y_train)
```

```
pred =
model.predict(x_test)
```

```
from     sklearn.discriminant_analysis
    import LinearDiscriminantAnalysis

lda = LinearDiscriminantAnalysis(n_components=2)
x_train  =  lda.fit_transform(x_train,  y_train)
x_test = lda.transform(x_test)
print("Accuracy", accuracy_score(y_test, pred))
print(confusion_matrix(y_test, pred))
```

```
Accuracy 0.9814814814814815
[[19 0 0]
 [ 1 20 0]
 [ 0 0 14]]
```