**Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**UCS2612 Machine Learning Laboratory**

**Assignment 9**

**Name:** V.S. Pranav **Reg No:** 3122 21 5001 069

**Applications of dimensionality reduction techniques**

**Github Link:** <https://github.com/vspr14/ml-lab-assn9>

1. 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.

**Code:**

## Importing the dataset

# Step 1: Importing and combining both datasets

# Import necessary libraries

import pandas as pd

import numpy as np

# Read the red wine dataset

red\_wine\_data = pd.read\_csv("winequality-red.csv", sep=";")

red\_wine\_data['type'] = 1 # Add a column 'type' with value 1 for red wines

# Read the white wine dataset

white\_wine\_data = pd.read\_csv("winequality-white.csv", sep=";")

white\_wine\_data['type'] = 0 # Add a column 'type' with value 0 for white wines

# Combine the datasets

wine\_data\_combined = pd.concat([red\_wine\_data, white\_wine\_data], ignore\_index=True)

# Display the first few rows of the combined dataset

print("Combined Wine Dataset:")

print(wine\_data\_combined.head())

## Pre-processing

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.impute import SimpleImputer

# Get all columns except 'type' as X

X = wine\_data\_combined.drop(columns=['type'])

# Encode non-numeric data into numeric

X\_encoded = pd.get\_dummies(X)

# Handle missing values by replacing them with the mean

imputer = SimpleImputer(strategy='mean')

X\_imputed = imputer.fit\_transform(X\_encoded)

# Perform normalization

scaler\_norm = MinMaxScaler()

X\_normalized = scaler\_norm.fit\_transform(X\_imputed)

# Perform standardization

scaler\_std = StandardScaler()

X\_final = scaler\_std.fit\_transform(X\_normalized)

# Convert the pre-processed data back to a DataFrame

X\_final\_df = pd.DataFrame(X\_final, columns=X\_encoded.columns)

# Display the pre-processed data

print("Pre-processed Data:")

print(X\_final\_df.head()) # Display first 5 rows

from scipy import stats

# Calculate z-scores for each column in X\_final\_df

threshold=2.0

z\_scores = np.abs(stats.zscore(X\_final\_df))

outlier\_indices = np.any(z\_scores > threshold, axis=1)

X\_final\_df = X\_final\_df[~outlier\_indices]

wine\_data\_combined = wine\_data\_combined[~outlier\_indices]

print("Shape of X\_final\_df:", X\_final\_df.shape)

print("Shape of wine\_data\_combined:", wine\_data\_combined.shape)

import matplotlib.pyplot as plt

import seaborn as sns

colors = ['lightblue', 'red']

plt.figure(figsize=(6, 6))

wine\_data\_combined['type'].value\_counts().plot(kind='pie', autopct='%1.1f%%', colors=colors)

plt.title('Proportion of White and Red Wines')

plt.xlabel('')

plt.ylabel('')

plt.legend(labels=['White wine', 'Red Wine'], loc='upper right') # Add legend

plt.show()

plt.figure(figsize=(10, 8))

sns.heatmap(X\_final\_df.corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Heatmap')

plt.show()

plt.figure(figsize=(8, 6))

sns.histplot(data=wine\_data\_combined, x='quality', hue='type', kde=True, bins=20, palette=colors)

plt.title('Quality Distribution for Red and White Wines')

plt.xlabel('Quality')

plt.ylabel('Count')

plt.legend(title='Wine Type', labels=['Red Wine', 'White Wine'])

plt.show()

from sklearn.decomposition import PCA

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA

num\_classes\_minus\_one = len(wine\_data\_combined['type'].unique()) - 1

pca = PCA(n\_components=num\_classes\_minus\_one)

X\_pca = pca.fit\_transform(X\_final\_df)

lda = LDA(n\_components=num\_classes\_minus\_one)

X\_lda = lda.fit\_transform(X\_final\_df, wine\_data\_combined['type'])

print("PCA Transformed Data Shape:", X\_pca.shape)

print("LDA Transformed Data Shape:", X\_lda.shape)

from sklearn.model\_selection import train\_test\_split

X\_pca\_train, X\_pca\_test, y\_train, y\_test = train\_test\_split(X\_pca, wine\_data\_combined['type'], test\_size=0.3, random\_state=42)

X\_lda\_train, X\_lda\_test, y\_train, y\_test = train\_test\_split(X\_lda, wine\_data\_combined['type'], test\_size=0.3, random\_state=42)

print("PCA Transformed Data - Training set shape:", X\_pca\_train.shape)

print("PCA Transformed Data - Testing set shape:", X\_pca\_test.shape)

print("LDA Transformed Data - Training set shape:", X\_lda\_train.shape)

print("LDA Transformed Data - Testing set shape:", X\_lda\_test.shape)

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

logreg\_pca = LogisticRegression()

logreg\_pca.fit(X\_pca\_train, y\_train)

logreg\_lda = LogisticRegression()

logreg\_lda.fit(X\_lda\_train, y\_train)

y\_pred\_pca = logreg\_pca.predict(X\_pca\_test)

y\_pred\_lda = logreg\_lda.predict(X\_lda\_test)

accuracy\_pca = accuracy\_score(y\_test, y\_pred\_pca)

accuracy\_lda = accuracy\_score(y\_test, y\_pred\_lda)

print("Accuracy using PCA transformed features:", accuracy\_pca)

print("Accuracy using LDA transformed features:", accuracy\_lda)

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

def plot\_confusion\_matrix(y\_true, y\_pred, title):

cm = confusion\_matrix(y\_true, y\_pred)

plt.figure(figsize=(4, 3))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.xlabel('Predicted labels')

plt.ylabel('True labels')

plt.title(title)

plt.show()

plot\_confusion\_matrix(y\_test, y\_pred\_pca, title='Confusion Matrix - PCA Transformed Features')

plot\_confusion\_matrix(y\_test, y\_pred\_lda, title='Confusion Matrix - LDA Transformed Features')

from sklearn.metrics import classification\_report

print("Classification Report - PCA Transformed Features:")

print(classification\_report(y\_test, y\_pred\_pca))

print("Classification Report - LDA Transformed Features:")

print(classification\_report(y\_test, y\_pred\_lda))

from sklearn.metrics import roc\_curve, auc

y\_proba\_pca\_train = logreg\_pca.predict\_proba(X\_pca\_train)[:, 1]

y\_proba\_lda\_train = logreg\_lda.predict\_proba(X\_lda\_train)[:, 1]

y\_proba\_pca\_test = logreg\_pca.predict\_proba(X\_pca\_test)[:, 1]

y\_proba\_lda\_test = logreg\_lda.predict\_proba(X\_lda\_test)[:, 1]

fpr\_pca\_train, tpr\_pca\_train, \_ = roc\_curve(y\_train, y\_proba\_pca\_train)

fpr\_lda\_train, tpr\_lda\_train, \_ = roc\_curve(y\_train, y\_proba\_lda\_train)

fpr\_pca\_test, tpr\_pca\_test, \_ = roc\_curve(y\_test, y\_proba\_pca\_test)

fpr\_lda\_test, tpr\_lda\_test, \_ = roc\_curve(y\_test, y\_proba\_lda\_test)

roc\_auc\_pca\_train = auc(fpr\_pca\_train, tpr\_pca\_train)

roc\_auc\_lda\_train = auc(fpr\_lda\_train, tpr\_lda\_train)

roc\_auc\_pca\_test = auc(fpr\_pca\_test, tpr\_pca\_test)

roc\_auc\_lda\_test = auc(fpr\_lda\_test, tpr\_lda\_test)

plt.figure(figsize=(8, 6))

plt.plot(fpr\_pca\_train, tpr\_pca\_train, color='blue', lw=2, label='ROC Curve - PCA Train (AUC = %0.5f)' % roc\_auc\_pca\_train)

plt.plot(fpr\_lda\_train, tpr\_lda\_train, color='red', lw=2, label='ROC Curve - LDA Train (AUC = %0.5f)' % roc\_auc\_lda\_train)

plt.plot(fpr\_pca\_test, tpr\_pca\_test, color='green', lw=2, label='ROC Curve - PCA Test (AUC = %0.5f)' % roc\_auc\_pca\_test)

plt.plot(fpr\_lda\_test, tpr\_lda\_test, color='orange', lw=2, label='ROC Curve - LDA Test (AUC = %0.5f)' % roc\_auc\_lda\_test)

plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC)')

plt.legend(loc="lower right")

plt.show()

**Sample Screenshots:**





