# PCA

#it is one feature extraction technique.

#we dont consider dependent variable that is why it is unsupervised model

# it will extract one which have most variance in detaset therefore we can reduce no of independent variable.

#it is basically classification problem it which ,here we will use logestic regression model and will apply PCA in classification

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

#----------------------------problem

#dependent variable is customer\_segment which have three type of wine and each wine corresponding to each customer

#each independent variable have different type of chemical for three type of customer

#we can not visulize lot of independent variable at once so we will apply dimensionlaity to limited variable which can explain maximum variance.

#----------------------------

# -------------Importing the dataset

#http://archive.ics.uci.edu/ml/datasets/wine dataset information

dataset = pd.read\_csv('Wine.csv')

X = dataset.iloc[:, 0:13].values

y = dataset.iloc[:, 13].values

# -----------Splitting the dataset into the Training set and Test 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)

# --------------Feature Scaling

#scaling must be apply when we are using dimensionality

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# -------------Applying PCA

#pca = PCA(n\_components = None) run with None then choose which explain most(sum too)

#PCA must apply after data preprocessing and before fitting the model in regression.

from sklearn.decomposition import PCA

pca = PCA(n\_components = 2)#extracted feature you want to get that explain most variance.

#put n\_components = None because first we need to look cumulative explain variance of different principal component

X\_train = pca.fit\_transform(X\_train)

X\_test = pca.transform(X\_test)

explained\_variance = pca.explained\_variance\_ratio\_#explain percentage of variance explained by each of principal component that extracted here.

#outcome of above is not original but it new created variable that explain percentage of variance

#if include 1 then 37% explained and include 2 then total 57% explained ,similarly other and last explain least variance

#for 2 component put n\_components=2 which explain 57% variance i.e good.

#after that reset console because X\_train,X\_test are not original because extracted from transform data but we want 2 column from original dataset so reset console i.e restart kernals then after select everything above from dimensiality code and followed by run.

#RESULLT::select 2 column explain maximum variance

# -------------Fitting Logistic Regression to the Training set

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

#diagonal present correct result

#accuracy(sum\_diagonal\_element/total no)

# Visualising the Training set results

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(('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()

# Visualising the Test set results

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(('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()