**Spring 2023: CS5710 – Machine Learning**

In-Class Programming Assignment-5

GitHub Link - <https://github.com/raimukul/MachineLearning_Assignments/tree/main/Assignment%2005>

Video link- <https://drive.google.com/file/d/13cmfNAbHYzUXgphNMItYNpi80udNLryJ/view?usp=share_link>

Code:

### 1. Principal Component Analysis

a. Apply PCA on CC dataset.

b. Apply k-means algorithm on the PCA result and report your observation if the silhouette score has

improved or not?

c. Perform Scaling+PCA+K-Means and report performance.

In [1]:

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

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

from sklearn.metrics import silhouette\_score

import warnings

warnings.filterwarnings('ignore')

*# read dataset*

df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Assignment 05/datasets/CC GENERAL.csv')

*# drop CUST\_ID column*

df.drop('CUST\_ID', axis=1, inplace=True)

*# drop rows with missing values*

df.dropna(inplace=True)

*# split dataset into train and test*

X\_train, X\_test = train\_test\_split(df, test\_size=0.2, random\_state=42)

*# scale fit training data*

scaler = StandardScaler()

scaler.fit(X\_train)

*# apply transform to training and test data*

X\_train = scaler.transform(X\_train)

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

*# Apply k-means algorithm on the original data*

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans.fit(X\_train)

y\_pred = kmeans.predict(X\_train)

sil\_original = silhouette\_score(X\_train, y\_pred)

print('Silhouette score for k-means on original data: ', sil\_original)

*# apply PCA to training and test data*

pca = PCA(n\_components=2)

pca.fit(X\_train)

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

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

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans.fit(X\_train)

y\_pred = kmeans.predict(X\_train)

sil\_pca = silhouette\_score(X\_train, y\_pred)

print('Silhouette score for k-means on PCA result: ', sil\_pca)

print('Silhouette score for k-means on original data is ', sil\_original, ' and silhouette score for k-means on PCA result is ', sil\_pca)

if(sil\_pca > sil\_original):

print('Silhouette score has improved')

else:

print('Silhouette score has not improved')

*# report performance on test data*

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

sil\_test = silhouette\_score(X\_test, y\_pred)

print('Silhouette score for k-means on test data: ', sil\_test)

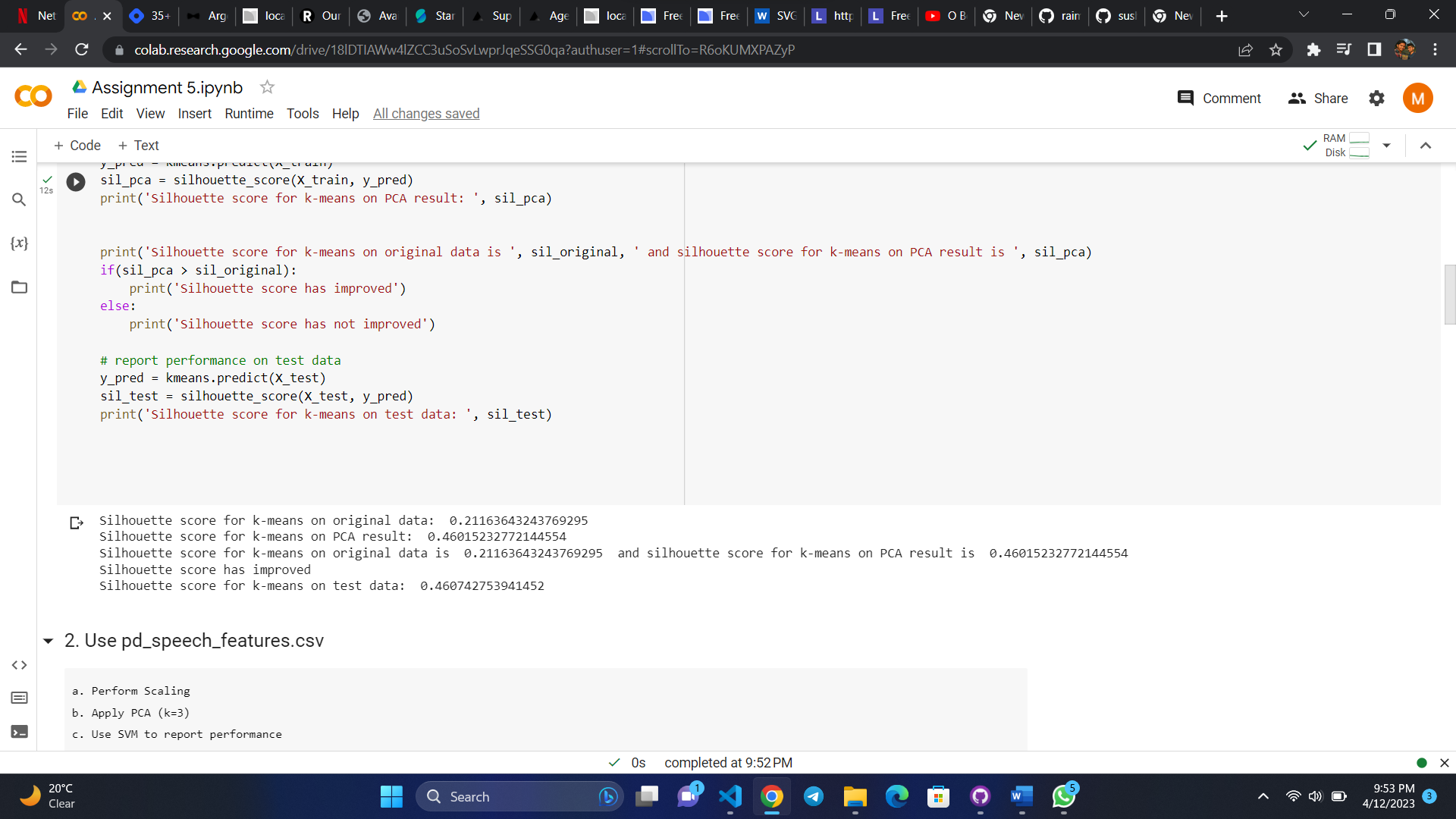
Silhouette score for k-means on original data: 0.21163643243769295

Silhouette score for k-means on PCA result: 0.46015232772144554

Silhouette score for k-means on original data is 0.21163643243769295 and silhouette score for k-means on PCA result is 0.46015232772144554

Silhouette score has improved

Silhouette score for k-means on test data: 0.460742753941452



### 2. Use pd\_speech\_features.csv

a. Perform Scaling

b. Apply PCA (k=3)

c. Use SVM to report performance

In [6]:

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

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

from sklearn.metrics import silhouette\_score

import warnings

warnings.filterwarnings('ignore')

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

*# Use pd\_speech\_features.csv*

df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Assignment 05/datasets/pd\_speech\_features.csv')

*# drop id column*

df.drop('id', axis=1, inplace=True)

*# drop rows with missing values*

df.dropna(inplace=True)

X = df.drop('class', axis=1)

y = df['class']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

*# a. Perform Scaling*

scaler = StandardScaler()

scaler.fit(X\_train)

*# apply transform to training and test data*

X\_train = scaler.transform(X\_train)

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

*# b. Apply PCA (k=3)*

pca = PCA(n\_components=3)

pca.fit(X\_train)

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

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

*# c. Use SVM to report performance*

svm = SVC()

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

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

print('Accuracy score: ', accuracy\_score(y\_test, y\_pred))

print('Confusion matrix: ', confusion\_matrix(y\_test, y\_pred))

print('Classification report: ', classification\_report(y\_test, y\_pred))

Accuracy score: 0.8026315789473685

Confusion matrix: [[ 16 22]

[ 8 106]]

Classification report: precision recall f1-score support

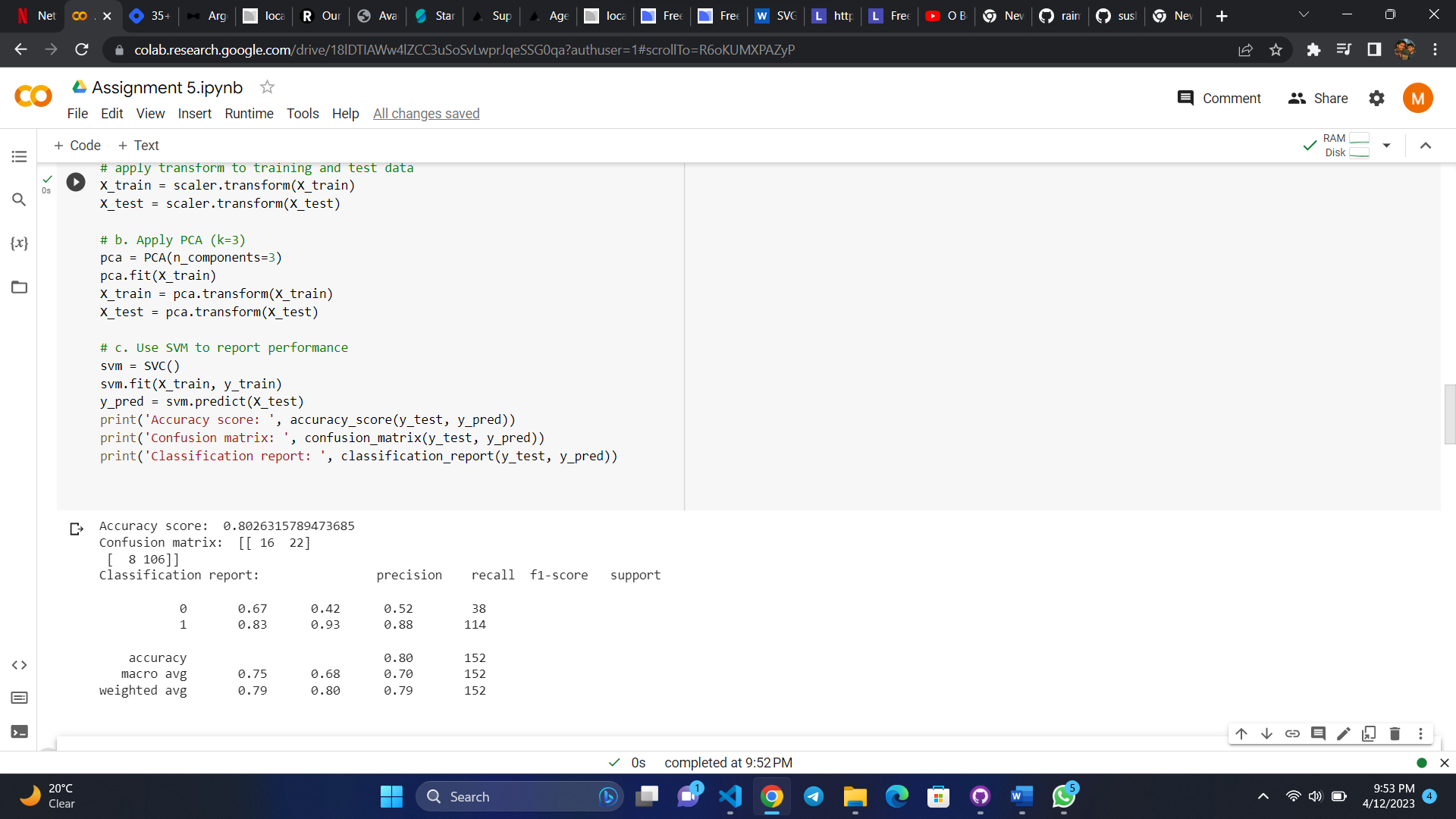
0 0.67 0.42 0.52 38

1 0.83 0.93 0.88 114

accuracy 0.80 152

macro avg 0.75 0.68 0.70 152

weighted avg 0.79 0.80 0.79 152



### 3. Apply Linear Discriminant Analysis (LDA) on Iris.csv dataset to reduce dimensionality of data tok=2.

In [5]:

import pandas as pd

import numpy as np

*# import lda*

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

*# read dataset*

df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Assignment 05/datasets/Iris.csv')

*# drop id column*

df.drop('Id', axis=1, inplace=True)

*# drop rows with missing values*

df.dropna(inplace=True)

*# split dataset into train and test*

X = df.drop('Species', axis=1)

y = df['Species']

*# apply LDA to training and test data*

lda = LinearDiscriminantAnalysis(n\_components=2)

lda.fit(X, y)

X = lda.transform(X)

print(X)

[[ 8.0849532 0.32845422]

[ 7.1471629 -0.75547326]

[ 7.51137789 -0.23807832]

[ 6.83767561 -0.64288476]

[ 8.15781367 0.54063935]

[ 7.72363087 1.48232345]

[ 7.23514662 0.3771537 ]

[ 7.62974497 0.01667246]

[ 6.58274132 -0.98737424]

[ 7.36884116 -0.91362729]

[ 8.42181434 0.67622968]

[ 7.24739721 -0.08292417]

[ 7.35062105 -1.0393597 ]

[ 7.59646896 -0.77671553]

[ 9.86936588 1.61486093]

[ 9.18033614 2.75558626]

[ 8.59760709 1.85442217]

[ 7.7995682 0.60905468]

[ 8.1000091 0.99610981]

[ 8.04543611 1.16244332]

[ 7.52046427 -0.156233 ]

[ 7.60526378 1.22757267]

[ 8.70408249 0.89959416]

[ 6.26374139 0.46023935]

[ 6.59191505 -0.36199821]

[ 6.79210164 -0.93823664]

[ 6.84048091 0.4848487 ]

[ 7.948386 0.23871551]

[ 8.01209273 0.11626909]

[ 6.85589572 -0.51715236]

[ 6.78303525 -0.72933749]

[ 7.38668238 0.59101728]

[ 9.16249492 1.25094169]

[ 9.49617185 1.84989586]

[ 7.36884116 -0.91362729]

[ 7.9756525 -0.13519572]

[ 8.63115466 0.4346228 ]

[ 7.36884116 -0.91362729]

[ 6.95602269 -0.67887846]

[ 7.71167183 0.01995843]

[ 7.9361354 0.69879338]

[ 5.6690533 -1.90328976]

[ 7.26559733 -0.24793625]

[ 6.42449823 1.26152073]

[ 6.88607488 1.07094506]

[ 6.77985104 -0.47815878]

[ 8.11232705 0.78881818]

[ 7.21095698 -0.33438897]

[ 8.33988749 0.6729437 ]

[ 7.69345171 -0.10577397]

[-1.45772244 0.04186554]

[-1.79768044 0.48879951]

[-2.41680973 -0.08234044]

[-2.26486771 -1.57609174]

[-2.55339693 -0.46282362]

[-2.41954768 -0.95728766]

[-2.44719309 0.79553574]

[-0.2160281 -1.57096512]

[-1.74591275 -0.80526746]

[-1.95838993 -0.35044011]

[-1.19023864 -2.61561292]

[-1.86140718 0.32050146]

[-1.15386577 -2.61693435]

[-2.65942607 -0.63412155]

[-0.38024071 0.09211958]

[-1.20280815 0.09561055]

[-2.7626699 0.03156949]

[-0.76227692 -1.63917546]

[-3.50940735 -1.6724835 ]

[-1.08410216 -1.6100398 ]

[-3.71895188 1.03509697]

[-0.99937 -0.47902036]

[-3.83709476 -1.39488292]

[-2.24344339 -1.41079358]

[-1.25428429 -0.53276537]

[-1.43952232 -0.12314653]

[-2.45921948 -0.91961551]

[-3.52471481 0.16379275]

[-2.58974981 -0.17075771]

[ 0.31197324 -1.29978446]

[-1.10232227 -1.7357722 ]

[-0.59844322 -1.92334798]

[-0.89605882 -0.89192518]

[-4.49567379 -0.87924754]

[-2.9265236 0.02499754]

[-2.10119821 1.18719828]

[-2.14367532 0.09713697]

[-2.48342912 -1.92190266]

[-1.31792367 -0.15753271]

[-1.95529307 -1.14514953]

[-2.38909697 -1.5823776 ]

[-2.28614469 -0.32562577]

[-1.26934019 -1.20042096]

[-0.28888857 -1.78315025]

[-2.00077969 -0.8969707 ]

[-1.16910587 -0.52787187]

[-1.6092782 -0.46274252]

[-1.41813799 -0.53933732]

[ 0.47271009 -0.78924756]

[-1.54557146 -0.58518894]

[-7.85608083 2.11161905]

[-5.5156825 -0.04401811]

[-6.30499392 0.46211638]

[-5.60355888 -0.34236987]

[-6.86344597 0.81602566]

[-7.42481805 -0.1726265 ]

[-4.68086447 -0.50758694]

[-6.31374875 -0.96068288]

[-6.33198886 -1.37715975]

[-6.87287126 2.69458147]

[-4.45364294 1.33693971]

[-5.4611095 -0.21035161]

[-5.67679825 0.82435717]

[-5.97407494 -0.10462115]

[-6.78782019 1.5744553 ]

[-5.82871291 1.98940576]

[-5.0664238 -0.02730214]

[-6.60847169 1.7420041 ]

[-9.18829265 -0.74909806]

[-4.76573133 -2.14417884]

[-6.29305487 1.63373692]

[-5.37314577 0.63153087]

[-7.58557489 -0.97390788]

[-4.38367513 -0.12213933]

[-5.73135125 1.28143515]

[-5.27583147 -0.0384815 ]

[-4.0923206 0.18307048]

[-4.08316687 0.51770204]

[-6.53257435 0.28724638]

[-4.577648 -0.84457527]

[-6.23500611 -0.70621819]

[-5.21836582 1.46644917]

[-6.81795935 0.56784684]

[-3.80972091 -0.93451896]

[-5.09023453 -2.11775698]

[-6.82119092 0.85698379]

[-6.54193229 2.41858841]

[-4.99356333 0.18488299]

[-3.94659967 0.60744074]

[-5.22159002 1.13613893]

[-6.67858684 1.785319 ]

[-5.13687786 1.97641389]

[-5.5156825 -0.04401811]

[-6.81196984 1.44440158]

[-6.87289126 2.40383699]

[-5.67401294 1.66134615]

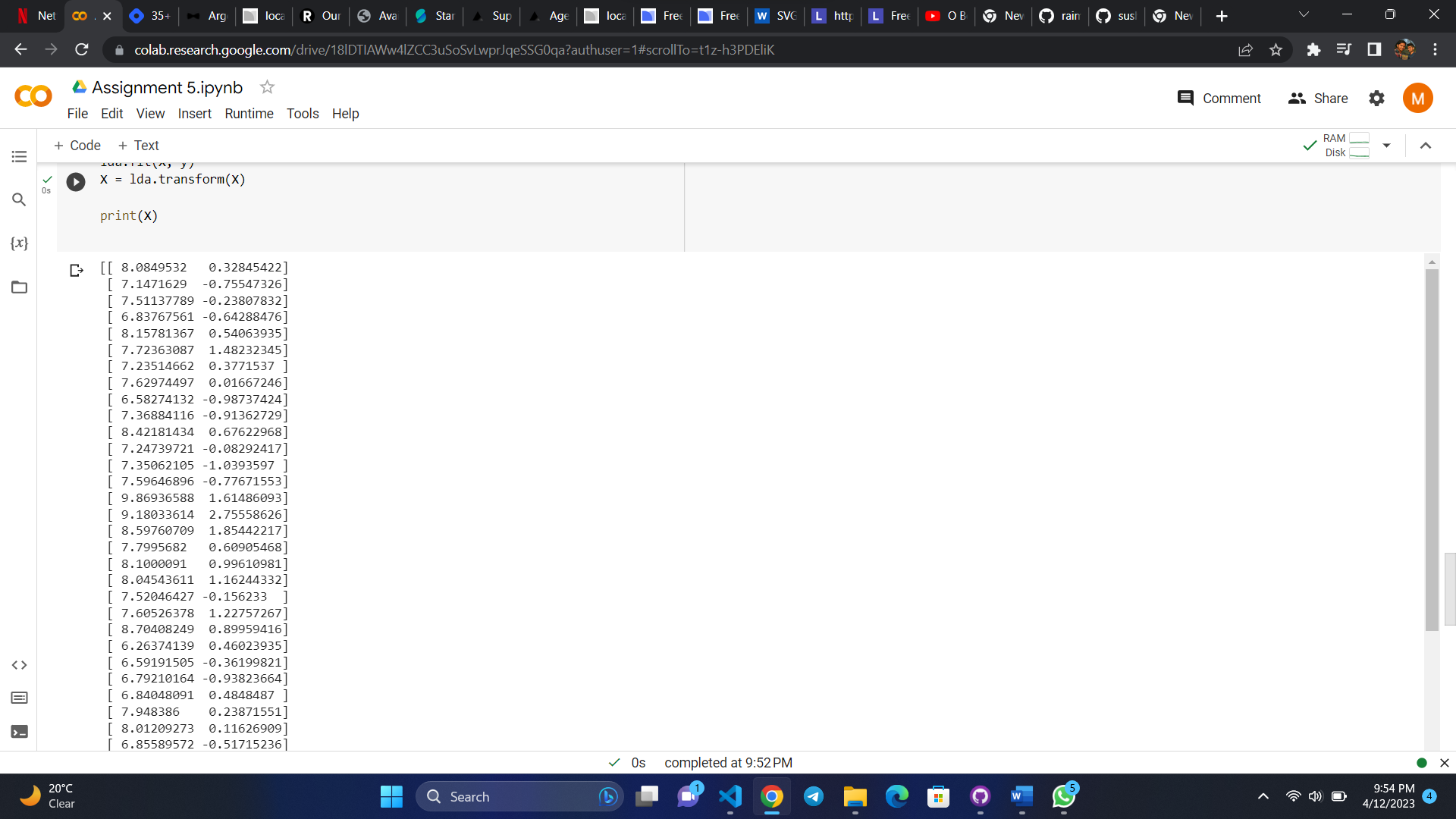
[-5.19712883 -0.36550576]

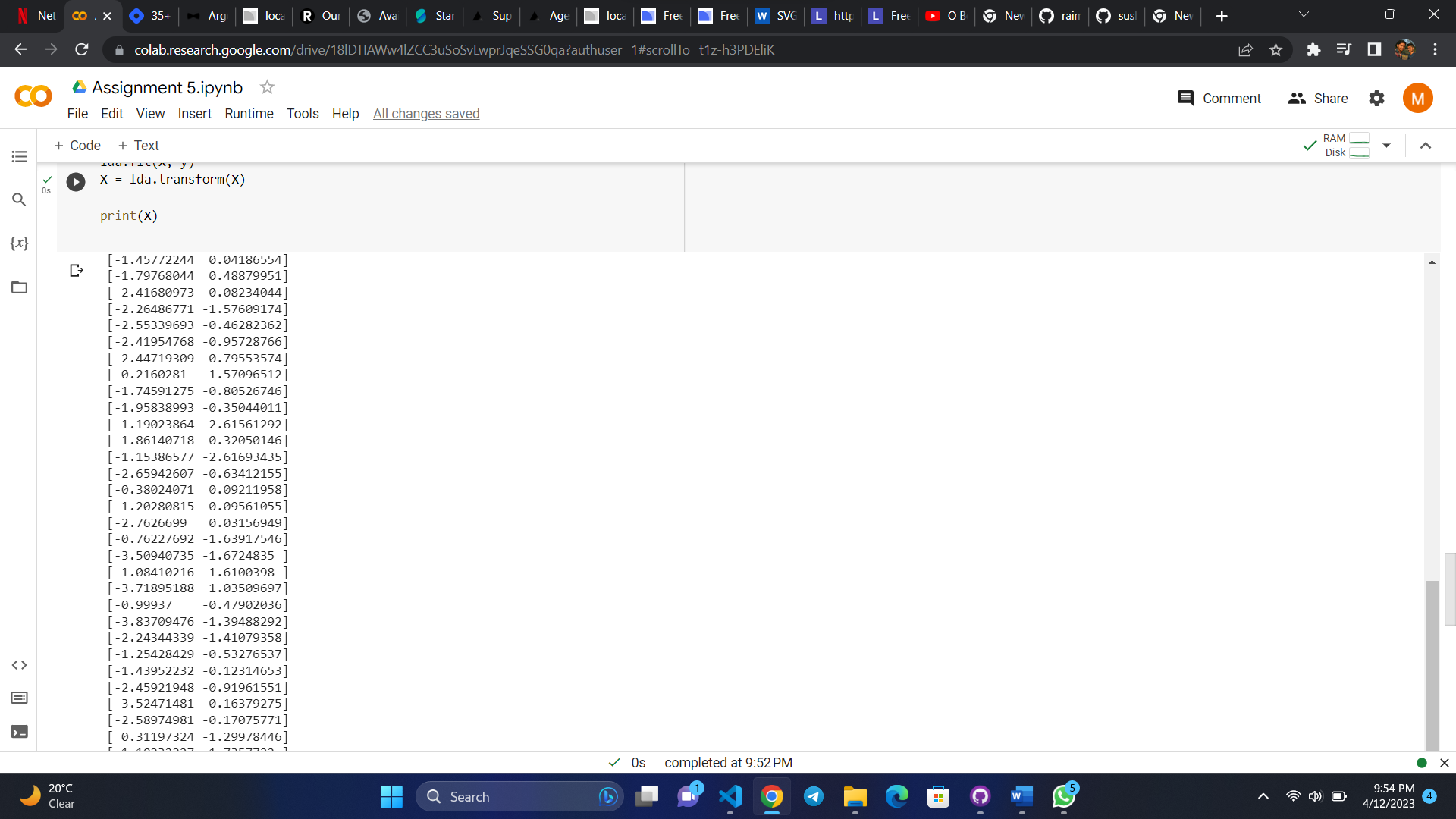
[-4.98171163 0.81297282]

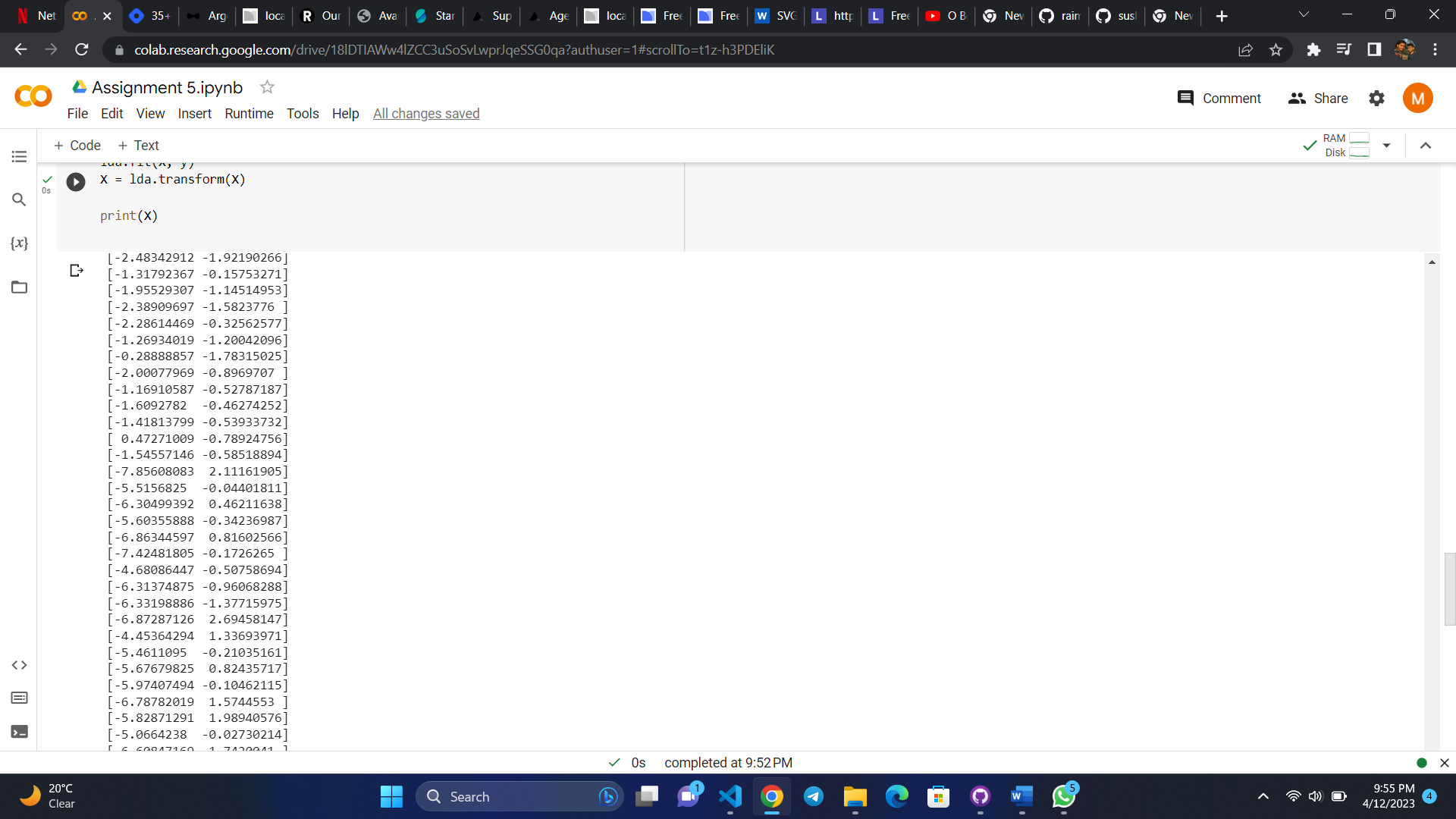
[-5.90148603 2.32075134]

[-4.68400868 0.32508073]]

Outputs:







### 4. Briefly identify the difference between PCA and LDA

**Answer -** PCA is an unsupervised algorithm that is used to reduce the dimensionality of the data. It is used to find the principal components of the data. LDA is also a supervised algorithm that is used to reduce the dimensionality of the data. It is used to find the linear combination of features that characterizes or separates two or more classes.