

Question 1:

`read_csv()` reads the csv file and stores in variable `df`

```
[1] # importing required libraries for assignment 5 here
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn import preprocessing, metrics
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
sns.set(style="white", color_codes=True)
import warnings
warnings.filterwarnings("ignore")

dataset_CC = pd.read_csv('./sample_data/CC.csv')
dataset_CC.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   CUST_ID                               8950 non-null   object
1   BALANCE                               8950 non-null   float64
2   BALANCE_FREQUENCY                     8950 non-null   float64
3   PURCHASES                             8950 non-null   float64
4   ONEOFF_PURCHASES                      8950 non-null   float64
5   INSTALLMENTS_PURCHASES                8950 non-null   float64
6   CASH_ADVANCE                          8950 non-null   float64
7   PURCHASES_FREQUENCY                   8950 non-null   float64
8   ONEOFF_PURCHASES_FREQUENCY            8950 non-null   float64
9   PURCHASES_INSTALLMENTS_FREQUENCY      8950 non-null   float64
10  CASH_ADVANCE_FREQUENCY                 8950 non-null   float64
```

`Fillna()` replaces all the null values with the mean value so that we donot have any NaN values in the dataframe

```
dataset_CC.head()
```

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166667	0.000000
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000000	0.000000
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000000	1.000000
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083333	0.083333
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083333	0.083333

```
[6] dataset_CC.isnull().any()
```

```
CUST_ID                               False
BALANCE                               False
BALANCE_FREQUENCY                     False
PURCHASES                             False
ONEOFF_PURCHASES                      False
INSTALLMENTS_PURCHASES                False
CASH_ADVANCE                          False
PURCHASES_FREQUENCY                   False
ONEOFF_PURCHASES_FREQUENCY            False
PURCHASES_INSTALLMENTS_FREQUENCY      False
CASH_ADVANCE_FREQUENCY                 False
CASH_ADVANCE_TRX                      False
PURCHASES_TRX                         False
CREDIT_LIMIT                          True
PAYMENTS                              False
MINIMUM_PAYMENTS                      True
PRC_FULL_PAYMENT                      False
```

The output of the PCA is fitted with `x` and stored in `x_pca`

jupyter Assignment 5 Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

In [40]: `# Apply PCA on CC Dataset`

```
pca = PCA(3)
x_pca = pca.fit_transform(x)
principalDf = pd.DataFrame(data = x_pca, columns = ['Principal Component 1', 'Principal Component 2', 'Principal Component 3'])
finalDf = pd.concat([principalDf, df.iloc[:, -1]], axis = 1)
finalDf.head()
```

Out[40]:

	Principal Component 1	Principal Component 2	Principal Component 3	TENURE
0	-4326.383979	921.566882	183.708383	12
1	4118.916665	-2432.846346	2369.969289	12
2	1497.907641	-1997.578694	-2125.631328	12
3	1394.548536	-1488.743453	-2431.799649	12
4	-3743.351896	757.342657	512.476492	12

In [41]: `# Applying K Means on PCA Result`

```
X = finalDf.iloc[:, 0:-1]
y = finalDf.iloc[:, -1]
```

In [42]:

```
nclusters = 3
km = KMeans(n_clusters=nclusters)
km.fit(X)
y_cluster_kmeans = km.predict(X)
```

Performed scaling and applied PCA

colab.research.google.com

Watch Naruto Online in HD with English Subbed, Dubbed

Watch Naruto Episode 73 for free on gogoanime

Untitled4.ipynb - Colaboratory

File Edit View Insert Runtime Tools Help Saving failed since 9:14 PM Comment Share Settings

+ Code + Text RAM Disk Editing

dataset_CC.fillna(dataset_CC.mean(), inplace=True)

dataset_CC.isnull().any()

CUST_ID False

BALANCE False

BALANCE_FREQUENCY False

PURCHASES False

ONEOFF_PURCHASES False

INSTALLMENTS_PURCHASES False

CASH_ADVANCE False

PURCHASES_FREQUENCY False

ONEOFF_PURCHASES_FREQUENCY False

PURCHASES_INSTALLMENTS_FREQUENCY False

CASH_ADVANCE_FREQUENCY False

CASH_ADVANCE_TRX False

PURCHASES_TRX False

CREDIT_LIMIT False

PAYMENTS False

MINIMUM_PAYMENTS False

PRC_FULL_PAYMENT False

TENURE False

dtype: bool

[8] x = dataset_CC.iloc[:, 1:-1]

y = dataset_CC.iloc[:, -1]

print(x.shape, y.shape)

(8950, 16) (8950,)

[9] pca = PCA(3)

x_pca = pca.fit_transform(x)

principalDf = pd.DataFrame(data = x_pca, columns = ['principal component 1', 'principal component 2', 'principal component 3'])

finalDf = pd.concat([principalDf, dataset_CC.iloc[:, -1]], axis = 1)

finalDf.head()

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

0s completed at 9:27 PM

```

[10] X = finalDf.iloc[:,0:-1]
     y = finalDf.iloc[:, -1]

nclusters = 3 # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X)

# predict the cluster for each data point
y_cluster_kmeans = km.predict(X)

# Summary of the predictions made by the classifier
print(classification_report(y, y_cluster_kmeans, zero_division=1))
print(confusion_matrix(y, y_cluster_kmeans))

train_accuracy = accuracy_score(y, y_cluster_kmeans)
print("\nAccuracy for our Training dataset with PCA:", train_accuracy)

#Calculate sihouette Score
score = metrics.silhouette_score(X, y_cluster_kmeans)
print("Sihouette Score: ",score)

```

	precision	recall	f1-score	support
0	0.00	1.00	0.00	0.0
1	0.00	1.00	0.00	0.0
2	0.00	1.00	0.00	0.0
6	1.00	0.00	0.00	204.0
7	1.00	0.00	0.00	190.0
8	1.00	0.00	0.00	196.0
9	1.00	0.00	0.00	175.0
10	1.00	0.00	0.00	236.0

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

0s completed at 9:27 PM

Question 2:

```

[20] dataset_pd = pd.read_csv('./sample_data/pd_speech_features.csv')
     dataset_pd.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 756 entries, 0 to 755
Columns: 755 entries, id to class
dtypes: float64(749), int64(6)
memory usage: 4.4 MB

dataset_pd.head()

```

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...	tqwt_kurtosisValue_dec_28	tqwt_kurt
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	...	1.5620	
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	...	1.5589	
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	...	1.5643	
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	...	3.7805	
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	...	6.1727	

5 rows x 755 columns

```

[22] dataset_pd.isnull().any()

id          False
gender      False
PPE         False
DFA         False
RPDE        False

```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

0s completed at 9:27 PM

Accuracy of svm is found using `accuracy_score()` and silhouette score is found using `silhouette_score()`

The screenshot shows a Google Colab notebook titled 'Untitled4.ipynb'. The code in cell [26] performs the following steps:

- Imports `train_test_split` from `sklearn.model_selection`.
- Splits the data into `X_train`, `X_test`, `y_train`, and `y_test` using `train_test_split(X, y, test_size=0.34, random_state=0)`.
- Imports `SVC` from `sklearn.svm`.
- Creates an `SVC` classifier, fits it to `X_train` and `y_train`, and predicts on `X_test`.
- Prints a summary of predictions using `classification_report`.
- Prints the confusion matrix using `confusion_matrix`.
- Calculates the accuracy score using `accuracy_score`.
- Calculates the silhouette score using `metrics.silhouette_score`.

The output of the code is as follows:

```
precision    recall  f1-score   support

0           0.67      0.42      0.51         62
1           0.84      0.93      0.88        196

 accuracy          0.75      0.68      0.81        258
 macro avg          0.75      0.68      0.70        258
weighted avg          0.80      0.81      0.79        258
```

Below the table, the accuracy and silhouette scores are printed:

```
[[ 26  36]
 [ 13 183]]
```

A message at the bottom states: "Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)"

Question 3:

Checking if there are any null values by calling `isnull()`

The screenshot shows a Google Colab notebook titled 'Untitled4.ipynb'. The code in cell [28] performs the following steps:

- Imports `LinearDiscriminantAnalysis` from `sklearn.discriminant_analysis`.
- Loads the Iris dataset using `pd.read_csv('./sample_data/Iris.csv')`.
- Prints the dataset information using `dataset_iris.info()`.

The output of `dataset_iris.info()` is:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype  
---  -
 0   Id                  150 non-null   int64  
 1   SepalLengthCm       150 non-null   float64
 2   SepalWidthCm        150 non-null   float64
 3   PetalLengthCm       150 non-null   float64
 4   PetalWidthCm        150 non-null   float64
 5   Species             150 non-null   object  
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In cell [29], the code checks for null values using `dataset_iris.isnull().any()`.

The output is:

```
Id                False
SepalLengthCm     False
SepalWidthCm      False
PetalLengthCm     False
PetalWidthCm      False
Species           False
dtype: bool
```

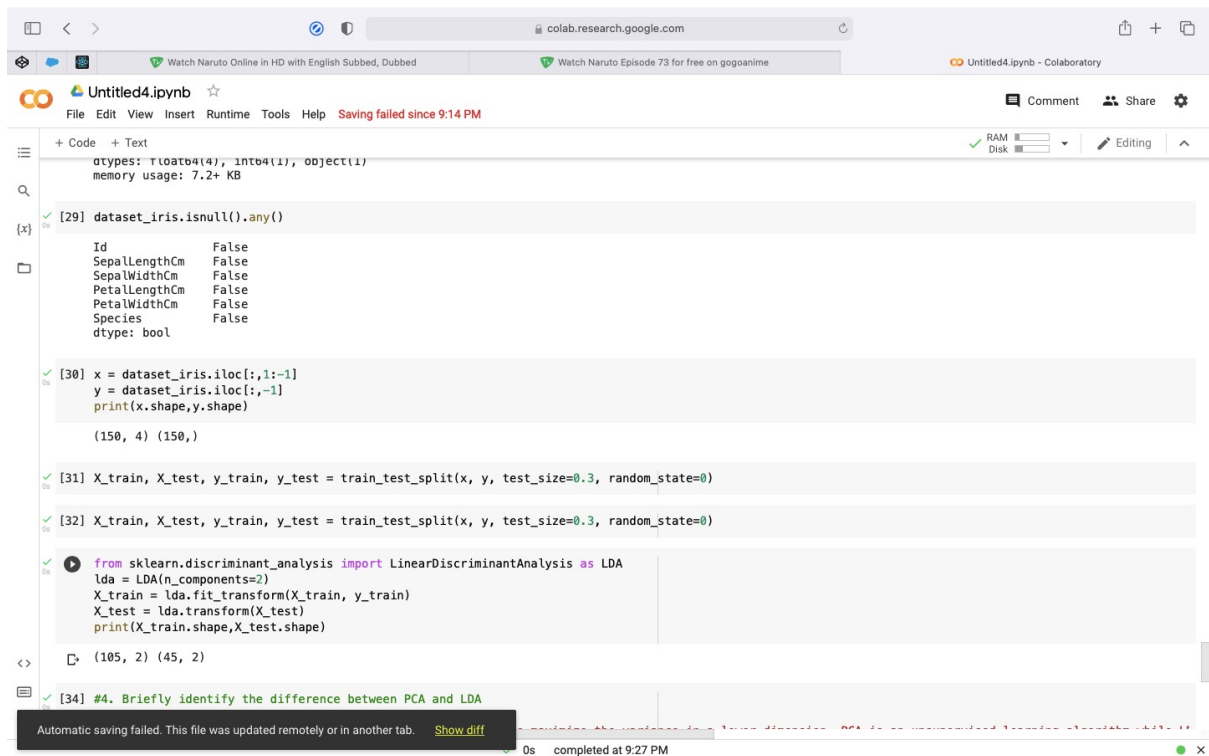
In cell [30], the code slices the data to exclude the last column and prints the shapes:

```
x = dataset_iris.iloc[:,1:-1]
y = dataset_iris.iloc[:, -1]
print(x.shape, y.shape)
```

The output is:

```
(150, 4) (150,)
```

A message at the bottom states: "Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)"



```
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB

[29] dataset_iris.isnull().any()

Id          False
SepalLengthCm  False
SepalWidthCm  False
PetalLengthCm  False
PetalWidthCm  False
Species      False
dtype: bool

[30] x = dataset_iris.iloc[:,1:-1]
y = dataset_iris.iloc[:, -1]
print(x.shape, y.shape)

(150, 4) (150,)

[31] X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)

[32] X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)

[33] from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components=2)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
print(X_train.shape, X_test.shape)

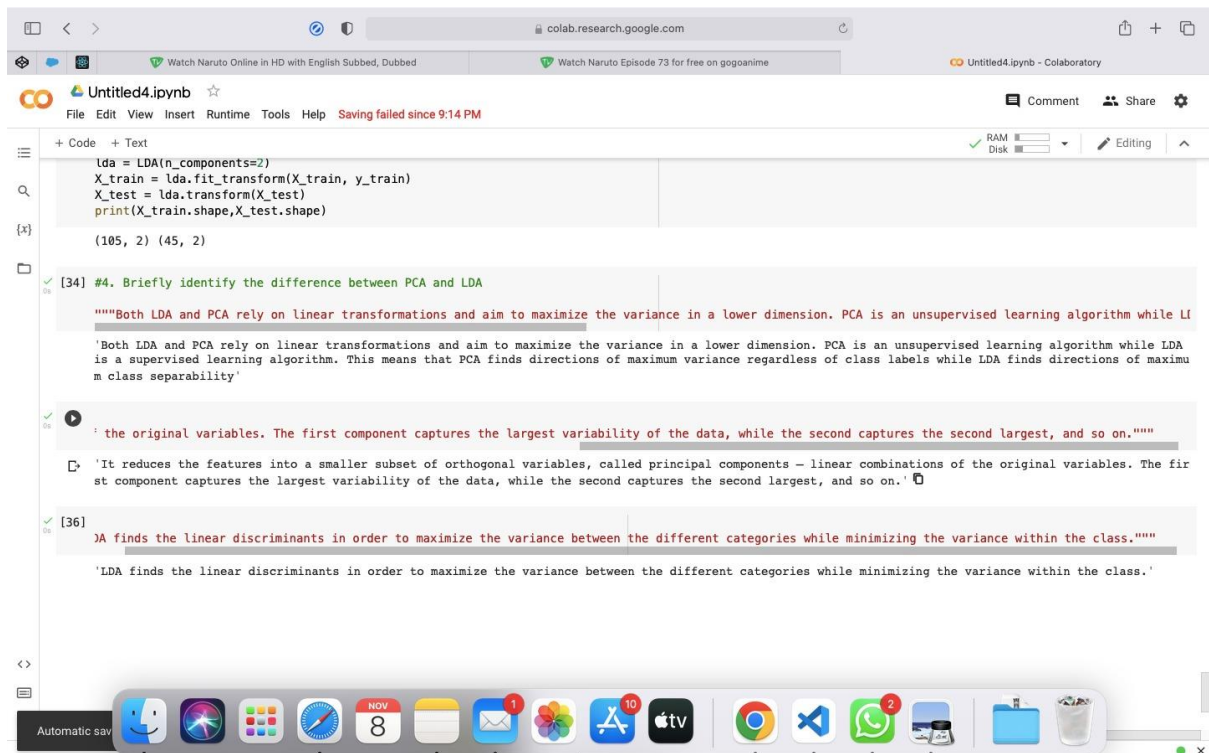
(105, 2) (45, 2)

[34] #4. Briefly identify the difference between PCA and LDA
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

0s completed at 9:27 PM

Question 4:



```
lda = LDA(n_components=2)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
print(X_train.shape, X_test.shape)

(105, 2) (45, 2)

[34] #4. Briefly identify the difference between PCA and LDA

"""Both LDA and PCA rely on linear transformations and aim to maximize the variance in a lower dimension. PCA is an unsupervised learning algorithm while LDA is a supervised learning algorithm. This means that PCA finds directions of maximum variance regardless of class labels while LDA finds directions of maximum class separability"""

[35] 'the original variables. The first component captures the largest variability of the data, while the second captures the second largest, and so on.'

[36] 'LDA finds the linear discriminants in order to maximize the variance between the different categories while minimizing the variance within the class.'

'LDA finds the linear discriminants in order to maximize the variance between the different categories while minimizing the variance within the class.'
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)