'''Problem Statement:

ICMR wants to analyze different types of cancers, such as breast cancer, renal cancer, colon cancer, lung cancer, and prostate cancer becoming a cause of worry in recent years. They would like to identify the probable cause of these cancers in terms of genes responsible for each cancer type. This would lead us to early identification of each type of cancer reducing the fatality rate.'''

'''The input dataset contains 802 samples for the corresponding 802 people who have been detected with different types of cancer. Each sample contains expression values of more than 20K genes. Samples have one of the types of tumors: BRCA, KIRC, COAD, LUAD, and PRAD.'''

BRCA cancer- breast cancer

COAD cancer- common malignant tumor in the digestive tract

KIRC cancer- Kidney renal

LUAD cancer- Lung cancer

PRAD cancer- Parotid tumors are abnormal growths of cells (tumors) that form in the parotid glands

```
# download the dataset
!wget
https://www.dropbox.com/sh/8q39v4rvo9hq7hy/AAAfAs9J12eevM_9_jPySJ1xa?
dl=0
```

```
--2022-09-17 09:20:49--
https://www.dropbox.com/sh/8q39v4rvo9hq7hy/AAAfAs9J12eevM_9_jPySJ1xa?
dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.85.18,
2620:100:6035:18::a27d:5512
Connecting to www.dropbox.com (www.dropbox.com)|162.125.85.18|:443...
connected.
HTTP request sent, awaiting response... 302 Found
Location: /sh/raw/8q39v4rvo9hq7hy/AAAfAs9J12eevM_9_jPySJ1xa
[following]
--2022-09-17 09:20:49--
https://www.dropbox.com/sh/raw/8q39v4rvo9hq7hy/AAAfAs9J12eevM_9_jPySJ1
xa
Reusing existing connection to www.dropbox.com:443.
```

```
HTTP request sent, awaiting response... 302 Found
Location:
https://uca09b971735544d5f2bfe2646b9.dl.dropboxusercontent.com/zip_dow
nload_get/BQdUyQq6BJOs2-bke5iDOBdVR2b406-
G_OKBubbR0OwZY_MDD9DRllkln2xhqywZAKRmABqWBtwRVNQ7Y5ygdgJKMS3c0QYC6uoTI
-c6sLs6KA# [following]
--2022-09-17 09:20:50--
https://uca09b971735544d5f2bfe2646b9.dl.dropboxusercontent.com/zip_dow
nload_get/BQdUyQq6BJOs2-bke5iDOBdVR2b406-
G_OKBubbR0OwZY_MDD9DRllkln2xhqywZAKRmABqWBtwRVNQ7Y5ygdgJKMS3c0QYC6uoTI
-c6sLs6KA
Resolving uca09b971735544d5f2bfe2646b9.dl.dropboxusercontent.com
(uca09b971735544d5f2bfe2646b9.dl.dropboxusercontent.com)...
162.125.7.15, 2620:100:6035:15::a27d:550f
Connecting to uca09b971735544d5f2bfe2646b9.dl.dropboxusercontent.com
(uca09b971735544d5f2bfe2646b9.dl.dropboxusercontent.com)|
162.125.7.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 206199237 (197M) [application/zip]
Saving to: ‘AAAfAs9J12eevM_9_jPySJ1xa?dl=0.1’

AAAfAs9J12eevM_9_jP 100%[===================>] 196.65M  18.9MB/s    in
12s

2022-09-17 09:21:03 (16.5 MB/s) - ‘AAAfAs9J12eevM_9_jPySJ1xa?dl=0.1’
saved [206199237/206199237]
```

```
# unzip the file
!unzip -qq /content/AAAfAs9J12eevM_9_jPySJ1xa?dl=0
```

```
warning:  stripped absolute path spec from /
mapname:  conversion of  failed
replace data.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
replace labels.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y

1 archive had fatal errors.
```

```
!pip install tensorflow==2.2.0
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow==2.2.0 in
/usr/local/lib/python3.7/dist-packages (2.2.0)
Requirement already satisfied: protobuf>=3.8.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(3.17.3)
Requirement already satisfied: wrapt>=1.11.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.14.1)
Requirement already satisfied: google-pasta>=0.1.8 in
```

```
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(0.2.0)
Requirement already satisfied: tensorboard<2.3.0,>=2.2.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(2.2.2)
Requirement already satisfied: h5py<2.11.0,>=2.10.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(2.10.0)
Requirement already satisfied: tensorflow-estimator<2.3.0,>=2.2.0
in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(2.2.0)
Requirement already satisfied: numpy<2.0,>=1.16.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.21.6)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.15.0)
Requirement already satisfied: wheel>=0.26 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(0.37.1)
Requirement already satisfied: scipy==1.4.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.4.1)
Requirement already satisfied: astunparse==1.6.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.6.3)
Requirement already satisfied: absl-py>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.2.0)
Requirement already satisfied: gast==0.3.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(0.3.3)
Requirement already satisfied: keras-preprocessing>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.1.2)
Requirement already satisfied: grpcio>=1.8.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.48.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(3.3.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.1.0)
Requirement already satisfied: google-auth<2,>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.35.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (0.4.6)
```

```
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (57.4.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.8.1)
Requirement already satisfied: werkzeug>=0.11.15 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.0.1)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (2.23.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.9)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (0.2.8)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.2.4)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
(1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in
/usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.12.0)
Requirement already satisfied: typing-extensions>=3.6.4 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
(4.1.1)
Requirement already satisfied: zipp>=0.5 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
(3.8.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<2,>=1.6.3->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
(0.4.8)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (2022.6.15)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in
```

```
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (2.10)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.3.0,>=2.2.0-
>tensorflow==2.2.0) (3.2.0)
```

```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
colors = ['royalblue','red','deeppink', 'maroon', 'mediumorchid',
'tan', 'forestgreen', 'olive', 'goldenrod', 'lightcyan', 'navy']
vectorizer = np.vectorize(lambda x: colors[x % len(colors)])

import warnings
warnings.filterwarnings(action='ignore',category=DeprecationWarning)
warnings.filterwarnings(action='ignore',category=FutureWarning)

import sys
import csv

csv.field_size_limit(sys.maxsize)
```

```
9223372036854775807
```

```
label = pd.read_csv('/content/labels.csv', delimiter=',',
engine='python')
data = pd.read_csv('/content/data.csv', delimiter=',',
engine='python')
data.describe()
```

```
           gene_0      gene_1      gene_2      gene_3      gene_4
gene_5  \
count  801.000000  801.000000  801.000000  801.000000  801.000000
801.0
mean     0.026642    3.010909    3.095350    6.722305    9.813612
0.0
std      0.136850    1.200828    1.065601    0.638819    0.506537
0.0
min      0.000000    0.000000    0.000000    5.009284    8.435999
0.0
25%      0.000000    2.299039    2.390365    6.303346    9.464466
```

```
0.0
50%      0.000000    3.143687    3.127006    6.655893    9.791599
0.0
75%      0.000000    3.883484    3.802534    7.038447   10.142324
0.0
max      1.482332    6.237034    6.063484   10.129528   11.355621
0.0

           gene_6      gene_7      gene_8      gene_9  ...  gene_20521
\
count  801.000000  801.000000  801.000000  801.000000  ...  801.000000

mean     7.405509    0.499882    0.016744    0.013428  ...    5.896573

std      1.108237    0.508799    0.133635    0.204722  ...    0.746399

min      3.930747    0.000000    0.000000    0.000000  ...    2.853517

25%      6.676042    0.000000    0.000000    0.000000  ...    5.454926

50%      7.450114    0.443076    0.000000    0.000000  ...    5.972582

75%      8.121984    0.789354    0.000000    0.000000  ...    6.411292

max     10.718190    2.779008    1.785592    4.067604  ...    7.771054


       gene_20522  gene_20523  gene_20524  gene_20525  gene_20526
gene_20527  \
count  801.000000  801.000000  801.000000  801.000000  801.000000
801.000000
mean     8.765891   10.056252    4.847727    9.741987   11.742228
10.155271
std      0.603176    0.379278    2.382728    0.533898    0.670371
0.580569
min      6.678368    8.669456    0.000000    7.974942    9.045255
7.530141
25%      8.383834    9.826027    3.130750    9.400747   11.315857
9.836525
50%      8.784144   10.066385    5.444935    9.784524   11.749802
10.191207
75%      9.147136   10.299025    6.637412   10.082269   12.177852
10.578561
max     11.105431   11.318243    9.207495   11.811632   13.715361
11.675653

       gene_20528  gene_20529  gene_20530
count  801.000000  801.000000  801.000000
mean     9.590726    5.528177    0.095411
```

```
std      0.563849    2.073859    0.364529
min      7.864533    0.593975    0.000000
25%      9.244219    4.092385    0.000000
50%      9.566511    5.218618    0.000000
75%      9.917888    6.876382    0.000000
max     12.813320   11.205836    5.254133

[8 rows x 20531 columns]
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 801 entries, 0 to 800
Columns: 20532 entries, Unnamed: 0 to gene_20530
dtypes: float64(20531), object(1)
memory usage: 125.5+ MB
```

Project Task: Week 1

Exploratory Data Analysis:

```
# Merge both the datasets
master_data = pd.merge(data, label)
master_data.head()
```

```
  Unnamed: 0  gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
\
0   sample_0     0.0  2.017209  3.265527  5.478487  10.431999     0.0

1   sample_1     0.0  0.592732  1.588421  7.586157   9.623011     0.0

2   sample_2     0.0  3.511759  4.327199  6.881787   9.870730     0.0

3   sample_3     0.0  3.663618  4.507649  6.659068  10.196184     0.0

4   sample_4     0.0  2.655741  2.821547  6.539454   9.738265     0.0


     gene_6    gene_7  gene_8  ...  gene_20522  gene_20523  gene_20524
\
0  7.175175  0.591871     0.0  ...    8.210257    9.723516    7.220030

1  6.816049  0.000000     0.0  ...    7.323865    9.740931    6.256586

2  6.972130  0.452595     0.0  ...    8.127123   10.908640    5.401607

3  7.843375  0.434882     0.0  ...    8.792959   10.141520    8.942805

4  6.566967  0.360982     0.0  ...    8.891425   10.373790    7.181162
```

```
   gene_20525  gene_20526  gene_20527  gene_20528  gene_20529
gene_20530  \
0    9.119813   12.003135    9.650743    8.921326    5.286759
0.0
1    8.381612   12.674552   10.517059    9.397854    2.094168
0.0
2    9.911597    9.045255    9.788359   10.090470    1.683023
0.0
3    9.601208   11.392682    9.694814    9.684365    3.292001
0.0
4    9.846910   11.922439    9.217749    9.461191    5.110372
0.0

   Class
0   PRAD
1   LUAD
2   PRAD
3   PRAD
4   BRCA

[5 rows x 20533 columns]
```

```
master_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 801 entries, 0 to 800
Columns: 20533 entries, Unnamed: 0 to Class
dtypes: float64(20531), object(2)
memory usage: 125.5+ MB
```

```
# check for null values
master_data.isnull().sum()
```

```
Unnamed: 0    0
gene_0        0
gene_1        0
gene_2        0
gene_3        0
             ..
gene_20527    0
gene_20528    0
gene_20529    0
gene_20530    0
Class         0
Length: 20533, dtype: int64
```

```
master_data.describe()
```

```
           gene_0      gene_1      gene_2      gene_3      gene_4
gene_5  \
```

```
count  801.000000  801.000000  801.000000  801.000000  801.000000
801.0
mean     0.026642    3.010909    3.095350    6.722305    9.813612
0.0
std      0.136850    1.200828    1.065601    0.638819    0.506537
0.0
min      0.000000    0.000000    0.000000    5.009284    8.435999
0.0
25%      0.000000    2.299039    2.390365    6.303346    9.464466
0.0
50%      0.000000    3.143687    3.127006    6.655893    9.791599
0.0
75%      0.000000    3.883484    3.802534    7.038447   10.142324
0.0
max      1.482332    6.237034    6.063484   10.129528   11.355621
0.0

           gene_6      gene_7      gene_8      gene_9  ...  gene_20521
\
count  801.000000  801.000000  801.000000  801.000000  ...  801.000000

mean     7.405509    0.499882    0.016744    0.013428  ...    5.896573

std      1.108237    0.508799    0.133635    0.204722  ...    0.746399

min      3.930747    0.000000    0.000000    0.000000  ...    2.853517

25%      6.676042    0.000000    0.000000    0.000000  ...    5.454926

50%      7.450114    0.443076    0.000000    0.000000  ...    5.972582

75%      8.121984    0.789354    0.000000    0.000000  ...    6.411292

max     10.718190    2.779008    1.785592    4.067604  ...    7.771054


       gene_20522  gene_20523  gene_20524  gene_20525  gene_20526
gene_20527  \
count  801.000000  801.000000  801.000000  801.000000  801.000000
801.000000
mean     8.765891   10.056252    4.847727    9.741987   11.742228
10.155271
std      0.603176    0.379278    2.382728    0.533898    0.670371
0.580569
min      6.678368    8.669456    0.000000    7.974942    9.045255
7.530141
25%      8.383834    9.826027    3.130750    9.400747   11.315857
9.836525
50%      8.784144   10.066385    5.444935    9.784524   11.749802
```

```
10.191207
75%      9.147136   10.299025    6.637412   10.082269   12.177852
10.578561
max     11.105431   11.318243    9.207495   11.811632   13.715361
11.675653

       gene_20528  gene_20529  gene_20530
count  801.000000  801.000000  801.000000
mean     9.590726    5.528177    0.095411
std      0.563849    2.073859    0.364529
min      7.864533    0.593975    0.000000
25%      9.244219    4.092385    0.000000
50%      9.566511    5.218618    0.000000
75%      9.917888    6.876382    0.000000
max     12.813320   11.205836    5.254133

[8 rows x 20531 columns]
```

Plot the merged dataset as a hierarchically-clustered heatmap

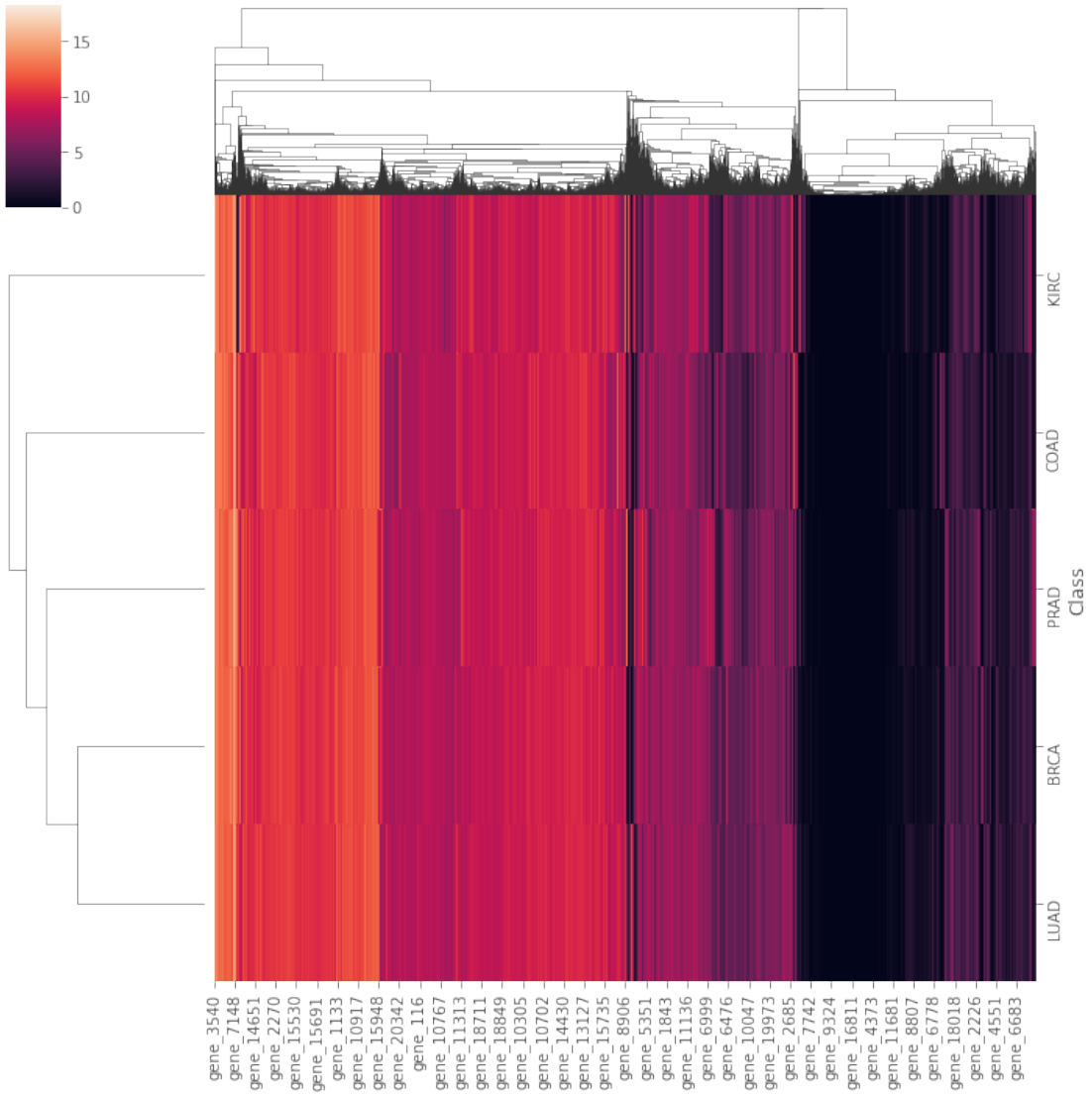
```
heatmap_data = pd.pivot_table(master_data, index=['Class'])

heatmap_data.head()
```

```
         gene_0    gene_1   gene_10   gene_100  gene_1000  gene_10000
\
Class

BRCA   0.011362  2.839739  0.544066  10.681488  10.303568    3.258028

COAD   0.022212  3.438381  0.357278  11.015745   9.951124    3.462039

KIRC   0.046544  2.398129  1.166824  10.238999  11.148094    1.651798

LUAD   0.041088  3.358260  0.607541  10.517670  10.503698    3.754181

PRAD   0.026544  3.441041  0.765608  10.282936   9.967433    1.949878


       gene_10001  gene_10002  gene_10003  gene_10004  ...  gene_9990
\
Class                                                  ...

BRCA     7.339461    7.900497    7.489146    7.508378  ...   1.969278

COAD     5.526673    7.487396    3.783493    6.959238  ...   2.216178

KIRC     6.895752    7.686932    7.269611    7.636246  ...   1.824964
```

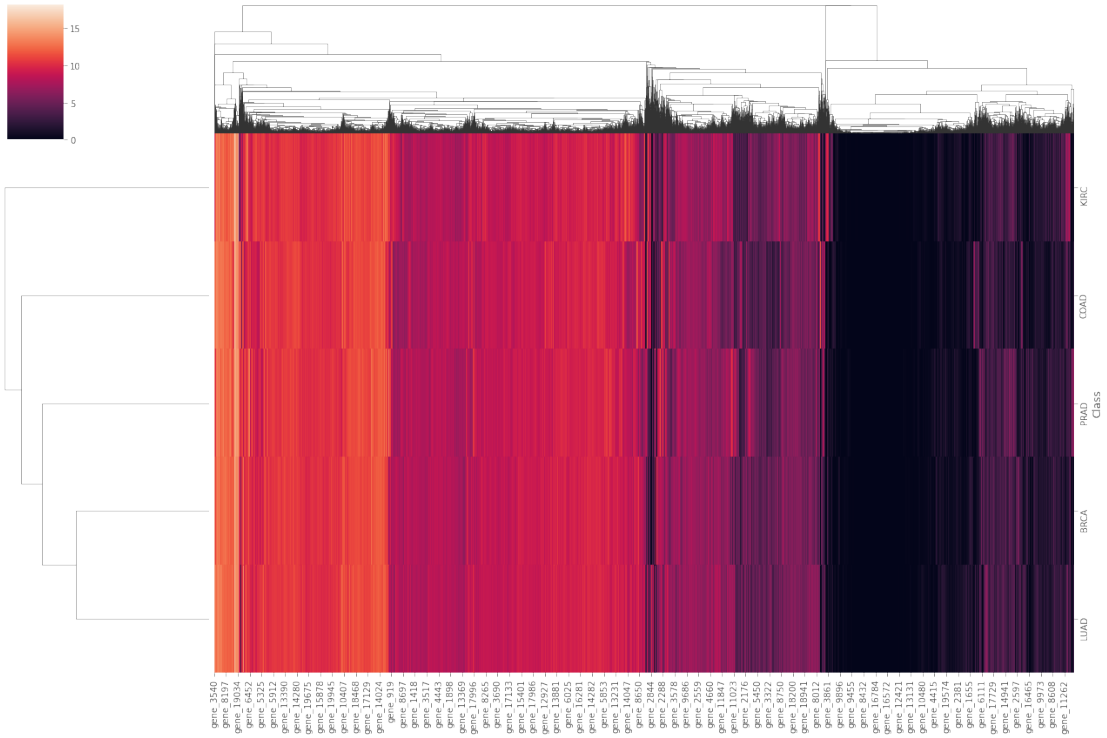
```
LUAD     7.281878    7.041924    6.145042    7.148682  ...   2.609490

PRAD     7.946141    8.529695    5.696368    7.396572  ...   1.623491


       gene_9991  gene_9992  gene_9993  gene_9994  gene_9995
gene_9996  \
Class

BRCA    5.142237   1.736160   2.312551   1.696127   2.493789
0.046527
COAD    0.354828   1.833606   1.619692   3.839205   2.396207
0.090327
KIRC    0.596508   2.393303   1.872888   1.289448   3.139623
0.130416
LUAD    2.801700   2.738326   1.869805   2.217144   2.459608
0.042070
PRAD    4.594215   1.684084   2.588050   1.703772   3.568490
0.572893

       gene_9997  gene_9998  gene_9999
Class
BRCA    2.099709   0.151063   6.954733
COAD    2.298246   0.065007   6.618466
KIRC    2.387948   0.148641   6.429343
LUAD    2.281828   0.056608   6.721517
PRAD    3.621548   0.094953   7.104225

[5 rows x 20531 columns]
```

```
sns.clustermap(heatmap_data)
plt.savefig('heatmap_with_Seaborn_clustermap_python.jpg',
            dpi=150, figsize=(8,12))
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/matrix.py:654:
UserWarning: Clustering large matrix with scipy. Installing
`fastcluster` may give better performance.
  warnings.warn(msg)
```




```
sns.clustermap(heatmap_data, figsize=(18,12))
plt.savefig('clustered_heatmap_with_dendrograms_Seaborn_clustermap_pyt
hon.jpg',dpi=150)
```



FEATURE SELECTION

split the data into train and test to avoid data-leakage

```
master_data.head(5)
```

```
  Unnamed: 0  gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
\
0   sample_0     0.0  2.017209  3.265527  5.478487  10.431999     0.0

1   sample_1     0.0  0.592732  1.588421  7.586157   9.623011     0.0

2   sample_2     0.0  3.511759  4.327199  6.881787   9.870730     0.0

3   sample_3     0.0  3.663618  4.507649  6.659068  10.196184     0.0

4   sample_4     0.0  2.655741  2.821547  6.539454   9.738265     0.0


     gene_6    gene_7  gene_8  ...  gene_20522  gene_20523  gene_20524
\
0  7.175175  0.591871     0.0  ...    8.210257    9.723516    7.220030

1  6.816049  0.000000     0.0  ...    7.323865    9.740931    6.256586

2  6.972130  0.452595     0.0  ...    8.127123   10.908640    5.401607
```

```
3  7.843375  0.434882     0.0  ...    8.792959   10.141520    8.942805

4  6.566967  0.360982     0.0  ...    8.891425   10.373790    7.181162


   gene_20525  gene_20526  gene_20527  gene_20528  gene_20529
gene_20530  \
0    9.119813   12.003135    9.650743    8.921326    5.286759
0.0
1    8.381612   12.674552   10.517059    9.397854    2.094168
0.0
2    9.911597    9.045255    9.788359   10.090470    1.683023
0.0
3    9.601208   11.392682    9.694814    9.684365    3.292001
0.0
4    9.846910   11.922439    9.217749    9.461191    5.110372
0.0

   Class
0   PRAD
1   LUAD
2   PRAD
3   PRAD
4   BRCA

[5 rows x 20533 columns]
```

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
X = master_data.iloc[:,1:-1]  #independent columns
y = master_data.iloc[:,-1]    #target column
```

```
X.head(5)
```

```
   gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
gene_6  \
0     0.0  2.017209  3.265527  5.478487  10.431999     0.0  7.175175

1     0.0  0.592732  1.588421  7.586157   9.623011     0.0  6.816049

2     0.0  3.511759  4.327199  6.881787   9.870730     0.0  6.972130

3     0.0  3.663618  4.507649  6.659068  10.196184     0.0  7.843375

4     0.0  2.655741  2.821547  6.539454   9.738265     0.0  6.566967


     gene_7  gene_8  gene_9  ...  gene_20521  gene_20522
gene_20523  \
0  0.591871     0.0     0.0  ...    4.926711    8.210257    9.723516
```

```
1  0.000000     0.0     0.0  ...    4.593372    7.323865    9.740931

2  0.452595     0.0     0.0  ...    5.125213    8.127123   10.908640

3  0.434882     0.0     0.0  ...    6.076566    8.792959   10.141520

4  0.360982     0.0     0.0  ...    5.996032    8.891425   10.373790


   gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
0    7.220030    9.119813   12.003135    9.650743    8.921326
5.286759
1    6.256586    8.381612   12.674552   10.517059    9.397854
2.094168
2    5.401607    9.911597    9.045255    9.788359   10.090470
1.683023
3    8.942805    9.601208   11.392682    9.694814    9.684365
3.292001
4    7.181162    9.846910   11.922439    9.217749    9.461191
5.110372

   gene_20530
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0

[5 rows x 20531 columns]
```

```
y.head(5)
```

```
0    PRAD
1    LUAD
2    PRAD
3    PRAD
4    BRCA
Name: Class, dtype: object
```

```
#apply SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)

dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)

#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
```

```
featureScores.columns = ['Specs','Score']  #naming the dataframe
columns
```

```
featureScores
```

```
            Specs      Score
0          gene_0   5.961875
1          gene_1  39.868223
2          gene_2  68.101756
3          gene_3  17.713982
4          gene_4   3.288049
...           ...        ...
20526  gene_20526   3.423439
20527  gene_20527   0.883221
20528  gene_20528   2.627112
20529  gene_20529  85.544690
20530  gene_20530  25.072727

[20531 rows x 2 columns]
```

```
print(featureScores.nlargest(10000,'Score'))  #print 10 best features
from 20531 features
```

```
            Specs        Score
9176    gene_9176  8580.566498
9175    gene_9175  7363.380532
15898  gene_15898  6902.074640
220      gene_220  6333.703681
219      gene_219  6259.461507
...           ...          ...
5653    gene_5653    27.130171
3993    gene_3993    27.129118
168      gene_168    27.127408
8281    gene_8281    27.100767
4090    gene_4090    27.097594

[10000 rows x 2 columns]
```

Feature Importance

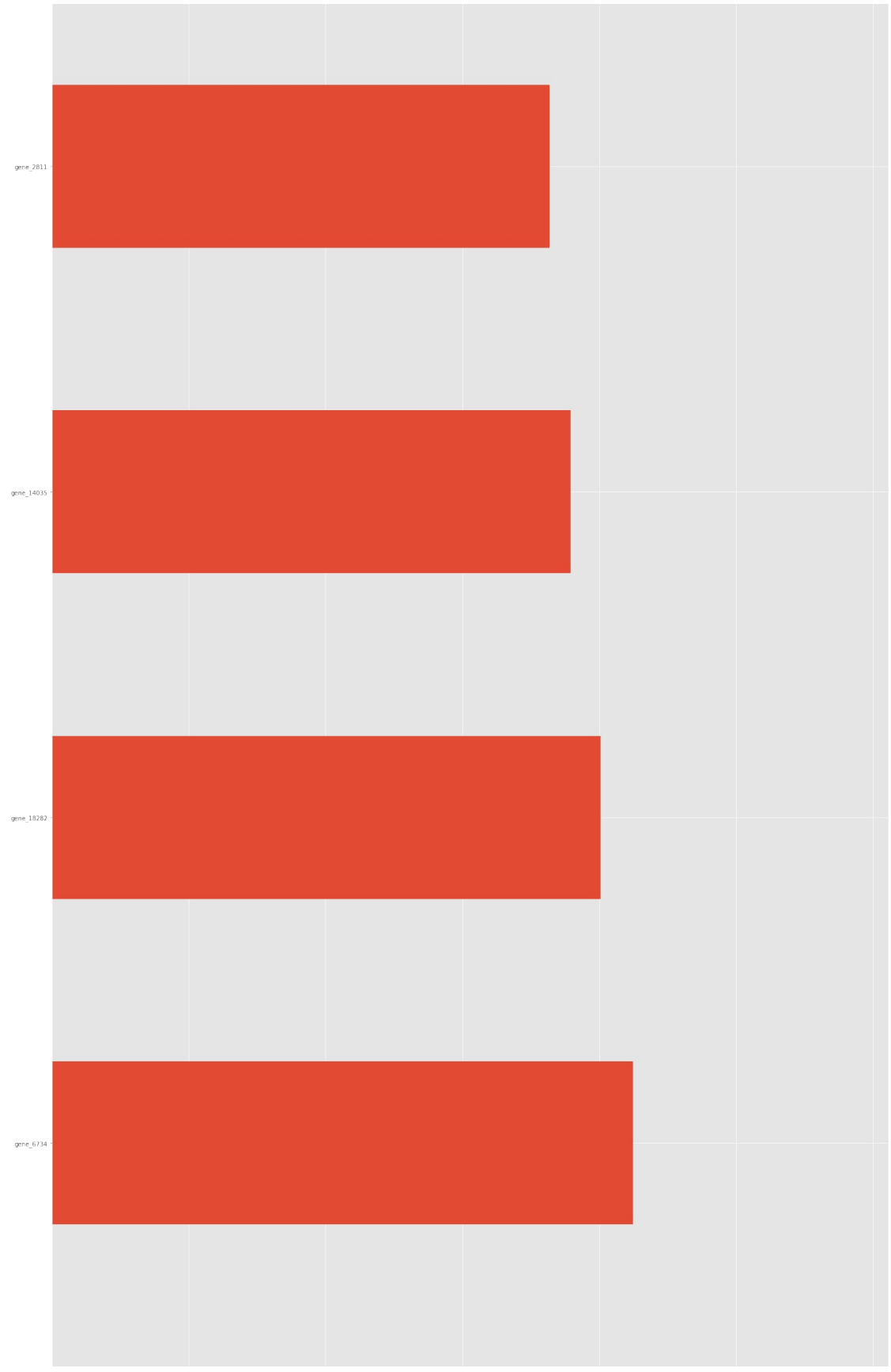
```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
```

```
ExtraTreesClassifier(bootstrap=False, class_weight=None,
criterion='gini',
                     max_depth=None, max_features='auto',
max_leaf_nodes=None,
                     min_impurity_decrease=0.0,
min_impurity_split=None,
```

```
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
n_jobs=None,
                       oob_score=False, random_state=None, verbose=0,
                       warm_start=False)

print(model.feature_importances_) #use inbuilt class
feature_importances of tree based classifiers
```

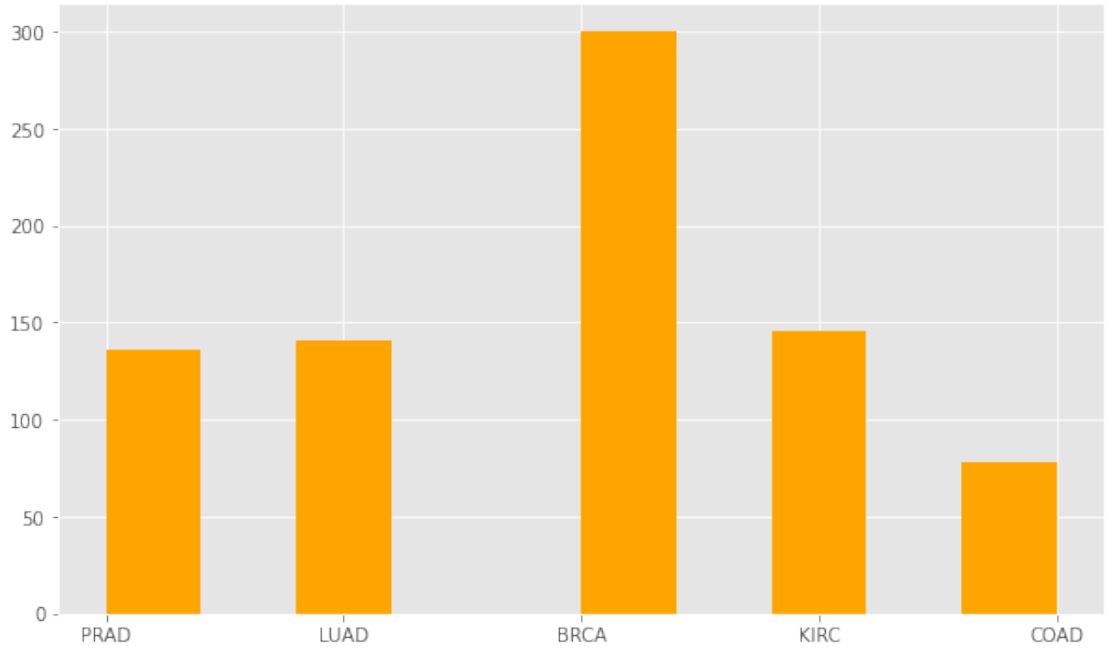
```
[0. 0. 0. ... 0. 0. 0.]
```

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_,
index=X.columns)
feat_importances.nlargest(10).plot(kind='barh', figsize=(25,100))
plt.show()
```


gene_2811
gene_14035
gene_18282
gene_6734

Checking histogram to check if the data is normally distributed

```
plt.figure(figsize=(10,6))
plt.hist(master_data['Class'], color = "orange")
plt.show()
```




```
non_cat_data = master_data.drop(['Unnamed: 0'], axis=1)
non_cat_data
```

```
     gene_0    gene_1    gene_2    gene_3     gene_4  gene_5    gene_6
\
0       0.0  2.017209  3.265527  5.478487  10.431999     0.0  7.175175

1       0.0  0.592732  1.588421  7.586157   9.623011     0.0  6.816049

2       0.0  3.511759  4.327199  6.881787   9.870730     0.0  6.972130

3       0.0  3.663618  4.507649  6.659068  10.196184     0.0  7.843375

4       0.0  2.655741  2.821547  6.539454   9.738265     0.0  6.566967

..      ...       ...       ...       ...        ...     ...       ...

796     0.0  1.865642  2.718197  7.350099  10.006003     0.0  6.764792

797     0.0  3.942955  4.453807  6.346597  10.056868     0.0  7.320331

798     0.0  3.249582  3.707492  8.185901   9.504082     0.0  7.536589

799     0.0  2.590339  2.787976  7.318624   9.987136     0.0  9.213464
```

```
800     0.0  2.325242  3.805932  6.530246   9.560367     0.0  7.957027


       gene_7  gene_8  gene_9  ...  gene_20522  gene_20523  gene_20524
\
0    0.591871     0.0     0.0  ...    8.210257    9.723516    7.220030

1    0.000000     0.0     0.0  ...    7.323865    9.740931    6.256586

2    0.452595     0.0     0.0  ...    8.127123   10.908640    5.401607

3    0.434882     0.0     0.0  ...    8.792959   10.141520    8.942805

4    0.360982     0.0     0.0  ...    8.891425   10.373790    7.181162

..        ...     ...     ...  ...         ...         ...         ...

796  0.496922     0.0     0.0  ...    9.118313   10.004852    4.484415

797  0.000000     0.0     0.0  ...    9.623335    9.823921    6.555327

798  1.811101     0.0     0.0  ...    8.610704   10.485517    3.589763

799  0.000000     0.0     0.0  ...    8.605387   11.004677    4.745888

800  0.000000     0.0     0.0  ...    8.594354   10.243079    9.139459


     gene_20525  gene_20526  gene_20527  gene_20528  gene_20529
gene_20530  \
0      9.119813   12.003135    9.650743    8.921326    5.286759
0.000000
1      8.381612   12.674552   10.517059    9.397854    2.094168
0.000000
2      9.911597    9.045255    9.788359   10.090470    1.683023
0.000000
3      9.601208   11.392682    9.694814    9.684365    3.292001
0.000000
4      9.846910   11.922439    9.217749    9.461191    5.110372
0.000000
..          ...         ...         ...         ...         ...
...
796    9.614701   12.031267    9.813063   10.092770    8.819269
0.000000
797    9.064002   11.633422   10.317266    8.745983    9.659081
0.000000
798    9.350636   12.180944   10.681194    9.466711    4.677458
0.586693
```

```
799    9.626383   11.198279   10.335513   10.400581    5.718751
0.000000
800   10.102934   11.641081   10.607358    9.844794    4.550716
0.000000

     Class
0     PRAD
1     LUAD
2     PRAD
3     PRAD
4     BRCA
..     ...
796   BRCA
797   LUAD
798   COAD
799   PRAD
800   PRAD

[801 rows x 20532 columns]
```

Checking for null values

```
nan_cols = [i for i in non_cat_data.columns if
non_cat_data[i].isnull().any()]
nan_cols
```

```
[]
```

```
# change the first column name
first_column = master_data.iloc[: , :1]
first_column.head(5)
```

```
  Unnamed: 0
0   sample_0
1   sample_1
2   sample_2
3   sample_3
4   sample_4
```

Dimensionality Reduction:

Each sample has expression values for around 20K genes. However, it may not be necessary to include all 20K genes expression values to analyze each cancer type. Therefore, we will identify a smaller set of attributes which will then be used to fit multiclass classification models. So, the first task targets the dimensionality reduction using various techniques such as, PCA, LDA, and t-SNE.bold text

Project Task: Week 2

Clustering Genes and Samples:

Our next goal is to identify groups of genes that behave similarly across samples and identify the distribution of samples corresponding to each cancer type. Therefore, this task focuses on applying various clustering techniques, e.g., k-means, hierarchical and mean shift clustering, on genes and samples.

First, apply the given clustering technique on all genes to identify:

Genes whose expression values are similar across all samples

Genes whose expression values are similar across samples of each cancer type

Next, apply the given clustering technique on all samples to identify:

Samples of the same class (cancer type) which also correspond to the same cluster

Samples identified to be belonging to another cluster but also to the same class (cancer type)

PRINCIPAL COMPONENT ANALYSIS (pca) - https://www.youtube.com/watch?v=FgakZw6K1QQ

REFERENCE -

https://www.youtube.com/watch?v=OFyyWcw2cyM

https://github.com/krishnaik06/Dimesnsionality-Reduction/blob/master/01-Principal%20Component%20Analysis.ipynb

PCA Visualization

As we've noticed before it is difficult to visualize high dimensional data, we can use PCA to find the first two principal components, and visualize the data in this new, two-dimensional space, with a single scatter-plot. Before we do this though, we'll need to scale our data so that each feature has a single unit variance.

```
from sklearn.preprocessing import StandardScaler

data_w_o_y = master_data.drop(['Unnamed: 0', 'Class'], axis=1)
data_w_o_y.head(5)
```

```
   gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
gene_6  \
0     0.0  2.017209  3.265527  5.478487  10.431999     0.0  7.175175

1     0.0  0.592732  1.588421  7.586157   9.623011     0.0  6.816049

2     0.0  3.511759  4.327199  6.881787   9.870730     0.0  6.972130

3     0.0  3.663618  4.507649  6.659068  10.196184     0.0  7.843375

4     0.0  2.655741  2.821547  6.539454   9.738265     0.0  6.566967
```

```
     gene_7  gene_8  gene_9  ...  gene_20521  gene_20522
gene_20523  \
0  0.591871     0.0     0.0  ...    4.926711    8.210257    9.723516

1  0.000000     0.0     0.0  ...    4.593372    7.323865    9.740931

2  0.452595     0.0     0.0  ...    5.125213    8.127123   10.908640

3  0.434882     0.0     0.0  ...    6.076566    8.792959   10.141520

4  0.360982     0.0     0.0  ...    5.996032    8.891425   10.373790


   gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
0    7.220030    9.119813   12.003135    9.650743    8.921326
5.286759
1    6.256586    8.381612   12.674552   10.517059    9.397854
2.094168
2    5.401607    9.911597    9.045255    9.788359   10.090470
1.683023
3    8.942805    9.601208   11.392682    9.694814    9.684365
3.292001
4    7.181162    9.846910   11.922439    9.217749    9.461191
5.110372

   gene_20530
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0

[5 rows x 20531 columns]
```

```
data_w_o_y.values.shape
```

```
(801, 20531)
```

```
scaler = StandardScaler()
X_Scaled = scaler.fit_transform(data_w_o_y)
X_Scaled
```

```
array([[-0.19479935, -0.82802988,  0.15980044, ..., -1.18793812,
        -0.11648251, -0.26190144],
       [-0.19479935, -2.01501735, -1.415042  , ..., -0.34227662,
        -1.65688871, -0.26190144],
       [-0.19479935,  0.41734754,  1.15673547, ...,  0.88686027,
        -1.85526414, -0.26190144],
```

```
       ...,
       [-0.19479935,  0.19888076,  0.57481583, ..., -0.22008186,
        -0.41046699,  1.3485582 ],
       [-0.19479935, -0.35045311, -0.28863152, ...,  1.43719268,
         0.09195083, -0.26190144],
       [-0.19479935, -0.57135218,  0.66725377, ...,  0.45087581,
        -0.47161901, -0.26190144]])
```

```
master_data.head(5)
```

```
  Unnamed: 0  gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
\
0   sample_0     0.0  2.017209  3.265527  5.478487  10.431999     0.0

1   sample_1     0.0  0.592732  1.588421  7.586157   9.623011     0.0

2   sample_2     0.0  3.511759  4.327199  6.881787   9.870730     0.0

3   sample_3     0.0  3.663618  4.507649  6.659068  10.196184     0.0

4   sample_4     0.0  2.655741  2.821547  6.539454   9.738265     0.0


     gene_6    gene_7  gene_8  ...  gene_20522  gene_20523  gene_20524
\
0  7.175175  0.591871     0.0  ...    8.210257    9.723516    7.220030

1  6.816049  0.000000     0.0  ...    7.323865    9.740931    6.256586

2  6.972130  0.452595     0.0  ...    8.127123   10.908640    5.401607

3  7.843375  0.434882     0.0  ...    8.792959   10.141520    8.942805

4  6.566967  0.360982     0.0  ...    8.891425   10.373790    7.181162


   gene_20525  gene_20526  gene_20527  gene_20528  gene_20529
gene_20530  \
0    9.119813   12.003135    9.650743    8.921326    5.286759
0.0
1    8.381612   12.674552   10.517059    9.397854    2.094168
0.0
2    9.911597    9.045255    9.788359   10.090470    1.683023
0.0
3    9.601208   11.392682    9.694814    9.684365    3.292001
0.0
4    9.846910   11.922439    9.217749    9.461191    5.110372
0.0
```

```
   Class
0   PRAD
1   LUAD
2   PRAD
3   PRAD
4   BRCA

[5 rows x 20533 columns]
```

master_data

```
     Unnamed: 0  gene_0    gene_1    gene_2    gene_3     gene_4
gene_5  \
0      sample_0     0.0  2.017209  3.265527  5.478487  10.431999
0.0
1      sample_1     0.0  0.592732  1.588421  7.586157   9.623011
0.0
2      sample_2     0.0  3.511759  4.327199  6.881787   9.870730
0.0
3      sample_3     0.0  3.663618  4.507649  6.659068  10.196184
0.0
4      sample_4     0.0  2.655741  2.821547  6.539454   9.738265
0.0
..          ...     ...       ...       ...       ...        ...     .
..
796  sample_796     0.0  1.865642  2.718197  7.350099  10.006003
0.0
797  sample_797     0.0  3.942955  4.453807  6.346597  10.056868
0.0
798  sample_798     0.0  3.249582  3.707492  8.185901   9.504082
0.0
799  sample_799     0.0  2.590339  2.787976  7.318624   9.987136
0.0
800  sample_800     0.0  2.325242  3.805932  6.530246   9.560367
0.0

       gene_6    gene_7  gene_8  ...  gene_20522  gene_20523
gene_20524  \
0    7.175175  0.591871     0.0  ...    8.210257    9.723516
7.220030
1    6.816049  0.000000     0.0  ...    7.323865    9.740931
6.256586
2    6.972130  0.452595     0.0  ...    8.127123   10.908640
5.401607
3    7.843375  0.434882     0.0  ...    8.792959   10.141520
8.942805
4    6.566967  0.360982     0.0  ...    8.891425   10.373790
7.181162
..        ...       ...     ...  ...         ...         ...         .
..
```

```
796  6.764792  0.496922     0.0  ...    9.118313   10.004852
4.484415
797  7.320331  0.000000     0.0  ...    9.623335    9.823921
6.555327
798  7.536589  1.811101     0.0  ...    8.610704   10.485517
3.589763
799  9.213464  0.000000     0.0  ...    8.605387   11.004677
4.745888
800  7.957027  0.000000     0.0  ...    8.594354   10.243079
9.139459

     gene_20525  gene_20526  gene_20527  gene_20528  gene_20529
gene_20530  \
0      9.119813   12.003135    9.650743    8.921326    5.286759
0.000000
1      8.381612   12.674552   10.517059    9.397854    2.094168
0.000000
2      9.911597    9.045255    9.788359   10.090470    1.683023
0.000000
3      9.601208   11.392682    9.694814    9.684365    3.292001
0.000000
4      9.846910   11.922439    9.217749    9.461191    5.110372
0.000000
..          ...         ...         ...         ...         ...
...
796    9.614701   12.031267    9.813063   10.092770    8.819269
0.000000
797    9.064002   11.633422   10.317266    8.745983    9.659081
0.000000
798    9.350636   12.180944   10.681194    9.466711    4.677458
0.586693
799    9.626383   11.198279   10.335513   10.400581    5.718751
0.000000
800   10.102934   11.641081   10.607358    9.844794    4.550716
0.000000

     Class
0     PRAD
1     LUAD
2     PRAD
3     PRAD
4     BRCA
..     ...
796   BRCA
797   LUAD
798   COAD
799   PRAD
800   PRAD

[801 rows x 20533 columns]
```

```
no_target_master_data = master_data.drop(['Class'], axis=1)
no_target_master_data.head(5)
```

```
  Unnamed: 0  gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
\
0   sample_0     0.0  2.017209  3.265527  5.478487  10.431999     0.0

1   sample_1     0.0  0.592732  1.588421  7.586157   9.623011     0.0

2   sample_2     0.0  3.511759  4.327199  6.881787   9.870730     0.0

3   sample_3     0.0  3.663618  4.507649  6.659068  10.196184     0.0

4   sample_4     0.0  2.655741  2.821547  6.539454   9.738265     0.0


     gene_6    gene_7  gene_8  ...  gene_20521  gene_20522  gene_20523
\
0  7.175175  0.591871     0.0  ...    4.926711    8.210257    9.723516

1  6.816049  0.000000     0.0  ...    4.593372    7.323865    9.740931

2  6.972130  0.452595     0.0  ...    5.125213    8.127123   10.908640

3  7.843375  0.434882     0.0  ...    6.076566    8.792959   10.141520

4  6.566967  0.360982     0.0  ...    5.996032    8.891425   10.373790


   gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
0    7.220030    9.119813   12.003135    9.650743    8.921326
5.286759
1    6.256586    8.381612   12.674552   10.517059    9.397854
2.094168
2    5.401607    9.911597    9.045255    9.788359   10.090470
1.683023
3    8.942805    9.601208   11.392682    9.694814    9.684365
3.292001
4    7.181162    9.846910   11.922439    9.217749    9.461191
5.110372

   gene_20530
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0
```

```
[5 rows x 20532 columns]
```

```
no_target_master_data.set_index('Unnamed: 0', inplace=True)
no_target_master_data
```

```
            gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
gene_6  \
Unnamed: 0

sample_0       0.0  2.017209  3.265527  5.478487  10.431999     0.0
7.175175
sample_1       0.0  0.592732  1.588421  7.586157   9.623011     0.0
6.816049
sample_2       0.0  3.511759  4.327199  6.881787   9.870730     0.0
6.972130
sample_3       0.0  3.663618  4.507649  6.659068  10.196184     0.0
7.843375
sample_4       0.0  2.655741  2.821547  6.539454   9.738265     0.0
6.566967
...            ...       ...       ...       ...        ...     ...
...
sample_796     0.0  1.865642  2.718197  7.350099  10.006003     0.0
6.764792
sample_797     0.0  3.942955  4.453807  6.346597  10.056868     0.0
7.320331
sample_798     0.0  3.249582  3.707492  8.185901   9.504082     0.0
7.536589
sample_799     0.0  2.590339  2.787976  7.318624   9.987136     0.0
9.213464
sample_800     0.0  2.325242  3.805932  6.530246   9.560367     0.0
7.957027

              gene_7  gene_8  gene_9  ...  gene_20521  gene_20522
gene_20523  \
Unnamed: 0                            ...

sample_0    0.591871     0.0     0.0  ...    4.926711    8.210257
9.723516
sample_1    0.000000     0.0     0.0  ...    4.593372    7.323865
9.740931
sample_2    0.452595     0.0     0.0  ...    5.125213    8.127123
10.908640
sample_3    0.434882     0.0     0.0  ...    6.076566    8.792959
10.141520
sample_4    0.360982     0.0     0.0  ...    5.996032    8.891425
10.373790
...              ...     ...     ...  ...         ...         ...
...
sample_796  0.496922     0.0     0.0  ...    6.088133    9.118313
```

```
10.004852
sample_797  0.000000     0.0     0.0  ...    6.371876    9.623335
9.823921
sample_798  1.811101     0.0     0.0  ...    5.719386    8.610704
10.485517
sample_799  0.000000     0.0     0.0  ...    5.785237    8.605387
11.004677
sample_800  0.000000     0.0     0.0  ...    6.403075    8.594354
10.243079

            gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
\
Unnamed: 0

sample_0      7.220030    9.119813   12.003135    9.650743    8.921326

sample_1      6.256586    8.381612   12.674552   10.517059    9.397854

sample_2      5.401607    9.911597    9.045255    9.788359   10.090470

sample_3      8.942805    9.601208   11.392682    9.694814    9.684365

sample_4      7.181162    9.846910   11.922439    9.217749    9.461191

...                ...         ...         ...         ...         ...

sample_796    4.484415    9.614701   12.031267    9.813063   10.092770

sample_797    6.555327    9.064002   11.633422   10.317266    8.745983

sample_798    3.589763    9.350636   12.180944   10.681194    9.466711

sample_799    4.745888    9.626383   11.198279   10.335513   10.400581

sample_800    9.139459   10.102934   11.641081   10.607358    9.844794


            gene_20529  gene_20530
Unnamed: 0
sample_0      5.286759    0.000000
sample_1      2.094168    0.000000
sample_2      1.683023    0.000000
sample_3      3.292001    0.000000
sample_4      5.110372    0.000000
...                ...         ...
sample_796    8.819269    0.000000
sample_797    9.659081    0.000000
sample_798    4.677458    0.586693
sample_799    5.718751    0.000000
```

```
sample_800    4.550716    0.000000

[801 rows x 20531 columns]
```

```
no_target_master_data.T
```

```
Unnamed: 0   sample_0   sample_1   sample_2   sample_3   sample_4
sample_5  \
gene_0       0.000000   0.000000   0.000000   0.000000   0.000000
0.000000
gene_1       2.017209   0.592732   3.511759   3.663618   2.655741
3.467853
gene_2       3.265527   1.588421   4.327199   4.507649   2.821547
3.581918
gene_3       5.478487   7.586157   6.881787   6.659068   6.539454
6.620243
gene_4      10.431999   9.623011   9.870730  10.196184   9.738265
9.706829
...               ...        ...        ...        ...        ...
...
gene_20526  12.003135  12.674552   9.045255  11.392682  11.922439
11.556995
gene_20527   9.650743  10.517059   9.788359   9.694814   9.217749
9.244150
gene_20528   8.921326   9.397854  10.090470   9.684365   9.461191
9.836473
gene_20529   5.286759   2.094168   1.683023   3.292001   5.110372
5.355133
gene_20530   0.000000   0.000000   0.000000   0.000000   0.000000
0.000000

Unnamed: 0   sample_6   sample_7   sample_8   sample_9  ...
sample_791  \
gene_0       0.000000   0.000000   0.000000   0.000000  ...
0.000000
gene_1       1.224966   2.854853   3.992125   3.642494  ...
3.080061
gene_2       1.691177   1.750478   2.772730   4.423558  ...
2.815739
gene_3       6.572007   7.226720   6.546692   6.849511  ...
6.209617
gene_4       9.640511   9.758691  10.488252   9.464466  ...
9.644469
...               ...        ...        ...        ...  ...         ..
.
gene_20526  13.256060  12.670377  12.498919  11.144295  ...
11.035335
gene_20527   9.664486   9.987733  10.389954   9.244851  ...
11.030377
gene_20528   9.244219   9.216872  10.390255   9.484299  ...
```

```
10.119304
gene_20529   8.330912   6.551490   7.828321   4.759151  ...
7.313500
gene_20530   0.000000   0.000000   0.000000   0.000000  ...
0.000000

Unnamed: 0  sample_792  sample_793  sample_794  sample_795  sample_796
\
gene_0        0.000000    0.000000    0.000000    0.436588    0.000000

gene_1        4.337404    2.068224    4.288388    4.472176    1.865642

gene_2        2.597126    0.857663    3.452490    4.908746    2.718197

gene_3        6.070379    6.218739    7.209151    5.937848    7.350099

gene_4        9.863990   10.623068    9.875620    9.330901   10.006003

...                ...         ...         ...         ...         ...

gene_20526   10.864960   10.703722   10.790014   10.961247   12.031267

gene_20527    9.617853    9.144204   10.698991   10.203226    9.813063

gene_20528   12.813320    9.682057   10.081523   10.030005   10.092770

gene_20529    3.805261    5.384968    4.376693    3.675703    8.819269

gene_20530    0.000000    0.789354    0.000000    0.000000    0.000000


Unnamed: 0  sample_797  sample_798  sample_799  sample_800
gene_0        0.000000    0.000000    0.000000    0.000000
gene_1        3.942955    3.249582    2.590339    2.325242
gene_2        4.453807    3.707492    2.787976    3.805932
gene_3        6.346597    8.185901    7.318624    6.530246
gene_4       10.056868    9.504082    9.987136    9.560367
...                ...         ...         ...         ...
gene_20526   11.633422   12.180944   11.198279   11.641081
gene_20527   10.317266   10.681194   10.335513   10.607358
gene_20528    8.745983    9.466711   10.400581    9.844794
gene_20529    9.659081    4.677458    5.718751    4.550716
gene_20530    0.000000    0.586693    0.000000    0.000000

[20531 rows x 801 columns]

from sklearn import preprocessing

# First center and scale the data
```

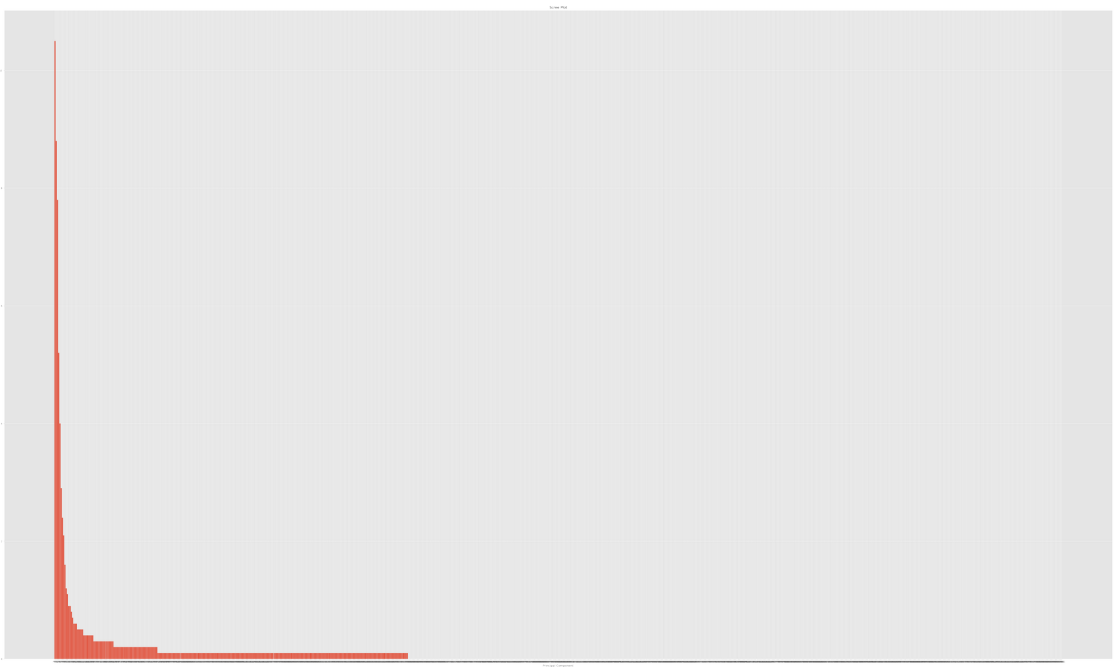
```
scaled_data = preprocessing.scale(no_target_master_data)

scaled_data
```

```
array([[-0.19479935, -0.82802988,  0.15980044, ..., -1.18793812,
        -0.11648251, -0.26190144],
       [-0.19479935, -2.01501735, -1.415042  , ..., -0.34227662,
        -1.65688871, -0.26190144],
       [-0.19479935,  0.41734754,  1.15673547, ...,  0.88686027,
        -1.85526414, -0.26190144],
       ...,
       [-0.19479935,  0.19888076,  0.57481583, ..., -0.22008186,
        -0.41046699,  1.3485582 ],
       [-0.19479935, -0.35045311, -0.28863152, ...,  1.43719268,
         0.09195083, -0.26190144],
       [-0.19479935, -0.57135218,  0.66725377, ...,  0.45087581,
        -0.47161901, -0.26190144]])
```

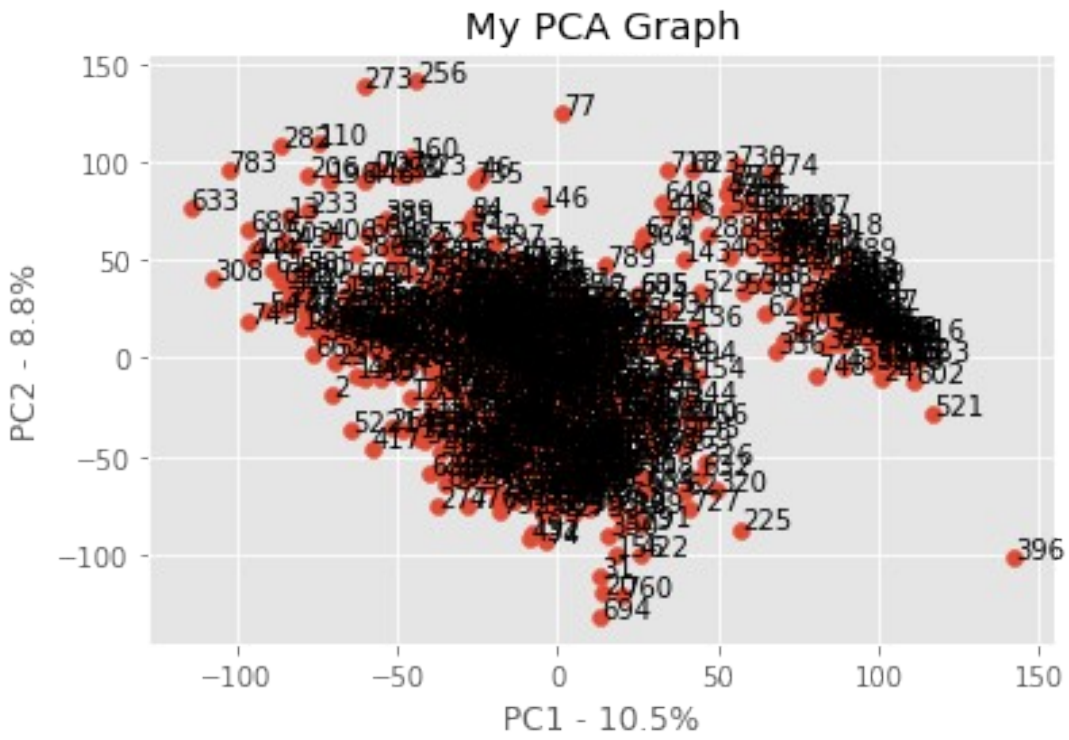
```
pca = PCA() # create a PCA object
pca.fit(scaled_data) # do the math
pca_data = pca.transform(scaled_data) # get PCA coordinates for
scaled_data
```

scree plot

```
#The following code constructs the Scree plot
per_var = np.round(pca.explained_variance_ratio_* 100, decimals=1)
labels = ['PC' + str(x) for x in range(1, len(per_var)+1)]

plt.figure(figsize=(80,48))
plt.bar(x=range(1,len(per_var)+1), height=per_var, tick_label=labels)
plt.ylabel('Percentage of Explained Variance')
plt.xlabel('Principal Component')
plt.title('Scree Plot')
plt.show()
```



```
#the following code makes a fancy looking plot using PC1 and PC2
pca_df = pd.DataFrame(pca_data, columns=labels)

plt.scatter(pca_df.PC1, pca_df.PC2)
plt.title('My PCA Graph')
plt.xlabel('PC1 - {0}%'.format(per_var[0]))
plt.ylabel('PC2 - {0}%'.format(per_var[1]))

for sample in pca_df.index:
    plt.annotate(sample, (pca_df.PC1.loc[sample],
pca_df.PC2.loc[sample]))

plt.show()
```




```
genes = no_target_master_data.columns.values.tolist()
genes
```

```
['gene_0',
 'gene_1',
 'gene_2',
 'gene_3',
 'gene_4',
 'gene_5',
 'gene_6',
 'gene_7',
 'gene_8',
 'gene_9',
 'gene_10',
 'gene_11',
 'gene_12',
 'gene_13',
 'gene_14',
 'gene_15',
 'gene_16',
 'gene_17',
 'gene_18',
 'gene_19',
 'gene_20',
 'gene_21',
 'gene_22',
 'gene_23',
```

```
'gene_24',
'gene_25',
'gene_26',
'gene_27',
'gene_28',
'gene_29',
'gene_30',
'gene_31',
'gene_32',
'gene_33',
'gene_34',
'gene_35',
'gene_36',
'gene_37',
'gene_38',
'gene_39',
'gene_40',
'gene_41',
'gene_42',
'gene_43',
'gene_44',
'gene_45',
'gene_46',
'gene_47',
'gene_48',
'gene_49',
'gene_50',
'gene_51',
'gene_52',
'gene_53',
'gene_54',
'gene_55',
'gene_56',
'gene_57',
'gene_58',
'gene_59',
'gene_60',
'gene_61',
'gene_62',
'gene_63',
'gene_64',
'gene_65',
'gene_66',
'gene_67',
'gene_68',
'gene_69',
'gene_70',
'gene_71',
'gene_72',
'gene_73',
```

```
'gene_74',
'gene_75',
'gene_76',
'gene_77',
'gene_78',
'gene_79',
'gene_80',
'gene_81',
'gene_82',
'gene_83',
'gene_84',
'gene_85',
'gene_86',
'gene_87',
'gene_88',
'gene_89',
'gene_90',
'gene_91',
'gene_92',
'gene_93',
'gene_94',
'gene_95',
'gene_96',
'gene_97',
'gene_98',
'gene_99',
'gene_100',
'gene_101',
'gene_102',
'gene_103',
'gene_104',
'gene_105',
'gene_106',
'gene_107',
'gene_108',
'gene_109',
'gene_110',
'gene_111',
'gene_112',
'gene_113',
'gene_114',
'gene_115',
'gene_116',
'gene_117',
'gene_118',
'gene_119',
'gene_120',
'gene_121',
'gene_122',
'gene_123',
```

'gene_124',
'gene_125',
'gene_126',
'gene_127',
'gene_128',
'gene_129',
'gene_130',
'gene_131',
'gene_132',
'gene_133',
'gene_134',
'gene_135',
'gene_136',
'gene_137',
'gene_138',
'gene_139',
'gene_140',
'gene_141',
'gene_142',
'gene_143',
'gene_144',
'gene_145',
'gene_146',
'gene_147',
'gene_148',
'gene_149',
'gene_150',
'gene_151',
'gene_152',
'gene_153',
'gene_154',
'gene_155',
'gene_156',
'gene_157',
'gene_158',
'gene_159',
'gene_160',
'gene_161',
'gene_162',
'gene_163',
'gene_164',
'gene_165',
'gene_166',
'gene_167',
'gene_168',
'gene_169',
'gene_170',
'gene_171',
'gene_172',
'gene_173',

'gene_174',
'gene_175',
'gene_176',
'gene_177',
'gene_178',
'gene_179',
'gene_180',
'gene_181',
'gene_182',
'gene_183',
'gene_184',
'gene_185',
'gene_186',
'gene_187',
'gene_188',
'gene_189',
'gene_190',
'gene_191',
'gene_192',
'gene_193',
'gene_194',
'gene_195',
'gene_196',
'gene_197',
'gene_198',
'gene_199',
'gene_200',
'gene_201',
'gene_202',
'gene_203',
'gene_204',
'gene_205',
'gene_206',
'gene_207',
'gene_208',
'gene_209',
'gene_210',
'gene_211',
'gene_212',
'gene_213',
'gene_214',
'gene_215',
'gene_216',
'gene_217',
'gene_218',
'gene_219',
'gene_220',
'gene_221',
'gene_222',
'gene_223',

'gene_224',
'gene_225',
'gene_226',
'gene_227',
'gene_228',
'gene_229',
'gene_230',
'gene_231',
'gene_232',
'gene_233',
'gene_234',
'gene_235',
'gene_236',
'gene_237',
'gene_238',
'gene_239',
'gene_240',
'gene_241',
'gene_242',
'gene_243',
'gene_244',
'gene_245',
'gene_246',
'gene_247',
'gene_248',
'gene_249',
'gene_250',
'gene_251',
'gene_252',
'gene_253',
'gene_254',
'gene_255',
'gene_256',
'gene_257',
'gene_258',
'gene_259',
'gene_260',
'gene_261',
'gene_262',
'gene_263',
'gene_264',
'gene_265',
'gene_266',
'gene_267',
'gene_268',
'gene_269',
'gene_270',
'gene_271',
'gene_272',
'gene_273',

'gene_274',
'gene_275',
'gene_276',
'gene_277',
'gene_278',
'gene_279',
'gene_280',
'gene_281',
'gene_282',
'gene_283',
'gene_284',
'gene_285',
'gene_286',
'gene_287',
'gene_288',
'gene_289',
'gene_290',
'gene_291',
'gene_292',
'gene_293',
'gene_294',
'gene_295',
'gene_296',
'gene_297',
'gene_298',
'gene_299',
'gene_300',
'gene_301',
'gene_302',
'gene_303',
'gene_304',
'gene_305',
'gene_306',
'gene_307',
'gene_308',
'gene_309',
'gene_310',
'gene_311',
'gene_312',
'gene_313',
'gene_314',
'gene_315',
'gene_316',
'gene_317',
'gene_318',
'gene_319',
'gene_320',
'gene_321',
'gene_322',
'gene_323',

```
'gene_324',
'gene_325',
'gene_326',
'gene_327',
'gene_328',
'gene_329',
'gene_330',
'gene_331',
'gene_332',
'gene_333',
'gene_334',
'gene_335',
'gene_336',
'gene_337',
'gene_338',
'gene_339',
'gene_340',
'gene_341',
'gene_342',
'gene_343',
'gene_344',
'gene_345',
'gene_346',
'gene_347',
'gene_348',
'gene_349',
'gene_350',
'gene_351',
'gene_352',
'gene_353',
'gene_354',
'gene_355',
'gene_356',
'gene_357',
'gene_358',
'gene_359',
'gene_360',
'gene_361',
'gene_362',
'gene_363',
'gene_364',
'gene_365',
'gene_366',
'gene_367',
'gene_368',
'gene_369',
'gene_370',
'gene_371',
'gene_372',
'gene_373',
```

'gene_374',
'gene_375',
'gene_376',
'gene_377',
'gene_378',
'gene_379',
'gene_380',
'gene_381',
'gene_382',
'gene_383',
'gene_384',
'gene_385',
'gene_386',
'gene_387',
'gene_388',
'gene_389',
'gene_390',
'gene_391',
'gene_392',
'gene_393',
'gene_394',
'gene_395',
'gene_396',
'gene_397',
'gene_398',
'gene_399',
'gene_400',
'gene_401',
'gene_402',
'gene_403',
'gene_404',
'gene_405',
'gene_406',
'gene_407',
'gene_408',
'gene_409',
'gene_410',
'gene_411',
'gene_412',
'gene_413',
'gene_414',
'gene_415',
'gene_416',
'gene_417',
'gene_418',
'gene_419',
'gene_420',
'gene_421',
'gene_422',
'gene_423',

'gene_424',
'gene_425',
'gene_426',
'gene_427',
'gene_428',
'gene_429',
'gene_430',
'gene_431',
'gene_432',
'gene_433',
'gene_434',
'gene_435',
'gene_436',
'gene_437',
'gene_438',
'gene_439',
'gene_440',
'gene_441',
'gene_442',
'gene_443',
'gene_444',
'gene_445',
'gene_446',
'gene_447',
'gene_448',
'gene_449',
'gene_450',
'gene_451',
'gene_452',
'gene_453',
'gene_454',
'gene_455',
'gene_456',
'gene_457',
'gene_458',
'gene_459',
'gene_460',
'gene_461',
'gene_462',
'gene_463',
'gene_464',
'gene_465',
'gene_466',
'gene_467',
'gene_468',
'gene_469',
'gene_470',
'gene_471',
'gene_472',
'gene_473',

'gene_474',
'gene_475',
'gene_476',
'gene_477',
'gene_478',
'gene_479',
'gene_480',
'gene_481',
'gene_482',
'gene_483',
'gene_484',
'gene_485',
'gene_486',
'gene_487',
'gene_488',
'gene_489',
'gene_490',
'gene_491',
'gene_492',
'gene_493',
'gene_494',
'gene_495',
'gene_496',
'gene_497',
'gene_498',
'gene_499',
'gene_500',
'gene_501',
'gene_502',
'gene_503',
'gene_504',
'gene_505',
'gene_506',
'gene_507',
'gene_508',
'gene_509',
'gene_510',
'gene_511',
'gene_512',
'gene_513',
'gene_514',
'gene_515',
'gene_516',
'gene_517',
'gene_518',
'gene_519',
'gene_520',
'gene_521',
'gene_522',
'gene_523',

'gene_524',
'gene_525',
'gene_526',
'gene_527',
'gene_528',
'gene_529',
'gene_530',
'gene_531',
'gene_532',
'gene_533',
'gene_534',
'gene_535',
'gene_536',
'gene_537',
'gene_538',
'gene_539',
'gene_540',
'gene_541',
'gene_542',
'gene_543',
'gene_544',
'gene_545',
'gene_546',
'gene_547',
'gene_548',
'gene_549',
'gene_550',
'gene_551',
'gene_552',
'gene_553',
'gene_554',
'gene_555',
'gene_556',
'gene_557',
'gene_558',
'gene_559',
'gene_560',
'gene_561',
'gene_562',
'gene_563',
'gene_564',
'gene_565',
'gene_566',
'gene_567',
'gene_568',
'gene_569',
'gene_570',
'gene_571',
'gene_572',
'gene_573',

'gene_574',
'gene_575',
'gene_576',
'gene_577',
'gene_578',
'gene_579',
'gene_580',
'gene_581',
'gene_582',
'gene_583',
'gene_584',
'gene_585',
'gene_586',
'gene_587',
'gene_588',
'gene_589',
'gene_590',
'gene_591',
'gene_592',
'gene_593',
'gene_594',
'gene_595',
'gene_596',
'gene_597',
'gene_598',
'gene_599',
'gene_600',
'gene_601',
'gene_602',
'gene_603',
'gene_604',
'gene_605',
'gene_606',
'gene_607',
'gene_608',
'gene_609',
'gene_610',
'gene_611',
'gene_612',
'gene_613',
'gene_614',
'gene_615',
'gene_616',
'gene_617',
'gene_618',
'gene_619',
'gene_620',
'gene_621',
'gene_622',
'gene_623',

'gene_624',
'gene_625',
'gene_626',
'gene_627',
'gene_628',
'gene_629',
'gene_630',
'gene_631',
'gene_632',
'gene_633',
'gene_634',
'gene_635',
'gene_636',
'gene_637',
'gene_638',
'gene_639',
'gene_640',
'gene_641',
'gene_642',
'gene_643',
'gene_644',
'gene_645',
'gene_646',
'gene_647',
'gene_648',
'gene_649',
'gene_650',
'gene_651',
'gene_652',
'gene_653',
'gene_654',
'gene_655',
'gene_656',
'gene_657',
'gene_658',
'gene_659',
'gene_660',
'gene_661',
'gene_662',
'gene_663',
'gene_664',
'gene_665',
'gene_666',
'gene_667',
'gene_668',
'gene_669',
'gene_670',
'gene_671',
'gene_672',
'gene_673',

'gene_674',
'gene_675',
'gene_676',
'gene_677',
'gene_678',
'gene_679',
'gene_680',
'gene_681',
'gene_682',
'gene_683',
'gene_684',
'gene_685',
'gene_686',
'gene_687',
'gene_688',
'gene_689',
'gene_690',
'gene_691',
'gene_692',
'gene_693',
'gene_694',
'gene_695',
'gene_696',
'gene_697',
'gene_698',
'gene_699',
'gene_700',
'gene_701',
'gene_702',
'gene_703',
'gene_704',
'gene_705',
'gene_706',
'gene_707',
'gene_708',
'gene_709',
'gene_710',
'gene_711',
'gene_712',
'gene_713',
'gene_714',
'gene_715',
'gene_716',
'gene_717',
'gene_718',
'gene_719',
'gene_720',
'gene_721',
'gene_722',
'gene_723',

```
'gene_724',
'gene_725',
'gene_726',
'gene_727',
'gene_728',
'gene_729',
'gene_730',
'gene_731',
'gene_732',
'gene_733',
'gene_734',
'gene_735',
'gene_736',
'gene_737',
'gene_738',
'gene_739',
'gene_740',
'gene_741',
'gene_742',
'gene_743',
'gene_744',
'gene_745',
'gene_746',
'gene_747',
'gene_748',
'gene_749',
'gene_750',
'gene_751',
'gene_752',
'gene_753',
'gene_754',
'gene_755',
'gene_756',
'gene_757',
'gene_758',
'gene_759',
'gene_760',
'gene_761',
'gene_762',
'gene_763',
'gene_764',
'gene_765',
'gene_766',
'gene_767',
'gene_768',
'gene_769',
'gene_770',
'gene_771',
'gene_772',
'gene_773',
```

'gene_774',
'gene_775',
'gene_776',
'gene_777',
'gene_778',
'gene_779',
'gene_780',
'gene_781',
'gene_782',
'gene_783',
'gene_784',
'gene_785',
'gene_786',
'gene_787',
'gene_788',
'gene_789',
'gene_790',
'gene_791',
'gene_792',
'gene_793',
'gene_794',
'gene_795',
'gene_796',
'gene_797',
'gene_798',
'gene_799',
'gene_800',
'gene_801',
'gene_802',
'gene_803',
'gene_804',
'gene_805',
'gene_806',
'gene_807',
'gene_808',
'gene_809',
'gene_810',
'gene_811',
'gene_812',
'gene_813',
'gene_814',
'gene_815',
'gene_816',
'gene_817',
'gene_818',
'gene_819',
'gene_820',
'gene_821',
'gene_822',
'gene_823',

```
'gene_824',
'gene_825',
'gene_826',
'gene_827',
'gene_828',
'gene_829',
'gene_830',
'gene_831',
'gene_832',
'gene_833',
'gene_834',
'gene_835',
'gene_836',
'gene_837',
'gene_838',
'gene_839',
'gene_840',
'gene_841',
'gene_842',
'gene_843',
'gene_844',
'gene_845',
'gene_846',
'gene_847',
'gene_848',
'gene_849',
'gene_850',
'gene_851',
'gene_852',
'gene_853',
'gene_854',
'gene_855',
'gene_856',
'gene_857',
'gene_858',
'gene_859',
'gene_860',
'gene_861',
'gene_862',
'gene_863',
'gene_864',
'gene_865',
'gene_866',
'gene_867',
'gene_868',
'gene_869',
'gene_870',
'gene_871',
'gene_872',
'gene_873',
```

```
'gene_874',
'gene_875',
'gene_876',
'gene_877',
'gene_878',
'gene_879',
'gene_880',
'gene_881',
'gene_882',
'gene_883',
'gene_884',
'gene_885',
'gene_886',
'gene_887',
'gene_888',
'gene_889',
'gene_890',
'gene_891',
'gene_892',
'gene_893',
'gene_894',
'gene_895',
'gene_896',
'gene_897',
'gene_898',
'gene_899',
'gene_900',
'gene_901',
'gene_902',
'gene_903',
'gene_904',
'gene_905',
'gene_906',
'gene_907',
'gene_908',
'gene_909',
'gene_910',
'gene_911',
'gene_912',
'gene_913',
'gene_914',
'gene_915',
'gene_916',
'gene_917',
'gene_918',
'gene_919',
'gene_920',
'gene_921',
'gene_922',
'gene_923',
```

'gene_924',
'gene_925',
'gene_926',
'gene_927',
'gene_928',
'gene_929',
'gene_930',
'gene_931',
'gene_932',
'gene_933',
'gene_934',
'gene_935',
'gene_936',
'gene_937',
'gene_938',
'gene_939',
'gene_940',
'gene_941',
'gene_942',
'gene_943',
'gene_944',
'gene_945',
'gene_946',
'gene_947',
'gene_948',
'gene_949',
'gene_950',
'gene_951',
'gene_952',
'gene_953',
'gene_954',
'gene_955',
'gene_956',
'gene_957',
'gene_958',
'gene_959',
'gene_960',
'gene_961',
'gene_962',
'gene_963',
'gene_964',
'gene_965',
'gene_966',
'gene_967',
'gene_968',
'gene_969',
'gene_970',
'gene_971',
'gene_972',
'gene_973',

```
 'gene_974',
 'gene_975',
 'gene_976',
 'gene_977',
 'gene_978',
 'gene_979',
 'gene_980',
 'gene_981',
 'gene_982',
 'gene_983',
 'gene_984',
 'gene_985',
 'gene_986',
 'gene_987',
 'gene_988',
 'gene_989',
 'gene_990',
 'gene_991',
 'gene_992',
 'gene_993',
 'gene_994',
 'gene_995',
 'gene_996',
 'gene_997',
 'gene_998',
 'gene_999',
 ...]
```

```
##########################
#
# Determine which genes had the biggest influence on PC1
#
##########################

## get the name of the top 10 measurements (genes) that contribute
## most to pc1.
## first, get the loading scores
loading_scores = pd.Series(pca.components_[0], index=genes)
## now sort the loading scores based on their magnitude
sorted_loading_scores =
loading_scores.abs().sort_values(ascending=False)

# get the names of the top 10 genes
top_10_genes = sorted_loading_scores[0:10].index.values

## print the gene names and their scores (and +/- sign)
print(loading_scores[top_10_genes])
```

```
gene_19862    0.019002
gene_17360    0.018985
gene_13489    0.018966
```

```
gene_15158    0.018777
gene_7031     0.018740
gene_7019     0.018657
gene_10788    0.018629
gene_13507   -0.018624
gene_6543     0.018595
gene_2288    -0.018592
dtype: float64
```

We can't choose top 10.

```
# Import PCA from sklearn and define the n_components as 2
from sklearn.decomposition import PCA
pca_with_2=PCA(n_components=2)
```

PCA with Scikit Learn uses a very similar process to other preprocessing functions that come with SciKit Learn. We instantiate a PCA object, find the principal components using the fit method, then apply the rotation and dimensionality reduction by calling transform().

We can also specify how many components we want to keep when creating the PCA object.

```
two_master_data = master_data

three_master_data = master_data

# Define data
df_pca = two_master_data.drop(['Unnamed: 0'], axis=1)
df_pca = df_pca.drop(['Class'], axis=1)
df_pca.head()
```

```
   gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
gene_6  \
0     0.0  2.017209  3.265527  5.478487  10.431999     0.0  7.175175

1     0.0  0.592732  1.588421  7.586157   9.623011     0.0  6.816049

2     0.0  3.511759  4.327199  6.881787   9.870730     0.0  6.972130

3     0.0  3.663618  4.507649  6.659068  10.196184     0.0  7.843375

4     0.0  2.655741  2.821547  6.539454   9.738265     0.0  6.566967


     gene_7  gene_8  gene_9  ...  gene_20521  gene_20522
gene_20523  \
0  0.591871     0.0     0.0  ...    4.926711    8.210257    9.723516

1  0.000000     0.0     0.0  ...    4.593372    7.323865    9.740931

2  0.452595     0.0     0.0  ...    5.125213    8.127123   10.908640
```

```
3  0.434882     0.0     0.0  ...    6.076566    8.792959   10.141520

4  0.360982     0.0     0.0  ...    5.996032    8.891425   10.373790


   gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
0    7.220030    9.119813   12.003135    9.650743    8.921326
5.286759
1    6.256586    8.381612   12.674552   10.517059    9.397854
2.094168
2    5.401607    9.911597    9.045255    9.788359   10.090470
1.683023
3    8.942805    9.601208   11.392682    9.694814    9.684365
3.292001
4    7.181162    9.846910   11.922439    9.217749    9.461191
5.110372

   gene_20530
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0

[5 rows x 20531 columns]

df_pca.values.shape

(801, 20531)

x_pca = df_pca.values

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_Scaled = scaler.fit_transform(x_pca)
X_Scaled

array([[-0.19479935, -0.82802988,  0.15980044, ..., -1.18793812,
        -0.11648251, -0.26190144],
       [-0.19479935, -2.01501735, -1.415042  , ..., -0.34227662,
        -1.65688871, -0.26190144],
       [-0.19479935,  0.41734754,  1.15673547, ...,  0.88686027,
        -1.85526414, -0.26190144],
       ...,
       [-0.19479935,  0.19888076,  0.57481583, ..., -0.22008186,
        -0.41046699,  1.3485582 ],
       [-0.19479935, -0.35045311, -0.28863152, ...,  1.43719268,
         0.09195083, -0.26190144],
       [-0.19479935, -0.57135218,  0.66725377, ...,  0.45087581,
        -0.47161901, -0.26190144]])
```

```
# Import PCA from sklearn and define the n_components as 2
from sklearn.decomposition import PCA
pca_with_2=PCA(n_components=2)

#Perform fit transform on the scaled data
X_pca_with_2 = pca_with_2.fit_transform(X_Scaled)
X_pca_with_2.shape
```

```
(801, 2)
```

```
X_pca_with_2
```

```
array([[-57.44698695,  95.41098124],
       [-16.91943006,   0.73246957],
       [-70.34521785, -19.30332741],
       ...,
       [ -4.13308964,  15.69001418],
       [-30.81475744,  33.52642257],
       [-22.34455668,   4.05235621]])
```

```
df_cat_data = three_master_data
```

```
df_cat_data.head(5)
```

```
  Unnamed: 0  gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
\
0   sample_0     0.0  2.017209  3.265527  5.478487  10.431999     0.0

1   sample_1     0.0  0.592732  1.588421  7.586157   9.623011     0.0

2   sample_2     0.0  3.511759  4.327199  6.881787   9.870730     0.0

3   sample_3     0.0  3.663618  4.507649  6.659068  10.196184     0.0

4   sample_4     0.0  2.655741  2.821547  6.539454   9.738265     0.0


     gene_6    gene_7  gene_8  ...  gene_20522  gene_20523  gene_20524
\
0  7.175175  0.591871     0.0  ...    8.210257    9.723516    7.220030

1  6.816049  0.000000     0.0  ...    7.323865    9.740931    6.256586

2  6.972130  0.452595     0.0  ...    8.127123   10.908640    5.401607

3  7.843375  0.434882     0.0  ...    8.792959   10.141520    8.942805

4  6.566967  0.360982     0.0  ...    8.891425   10.373790    7.181162
```

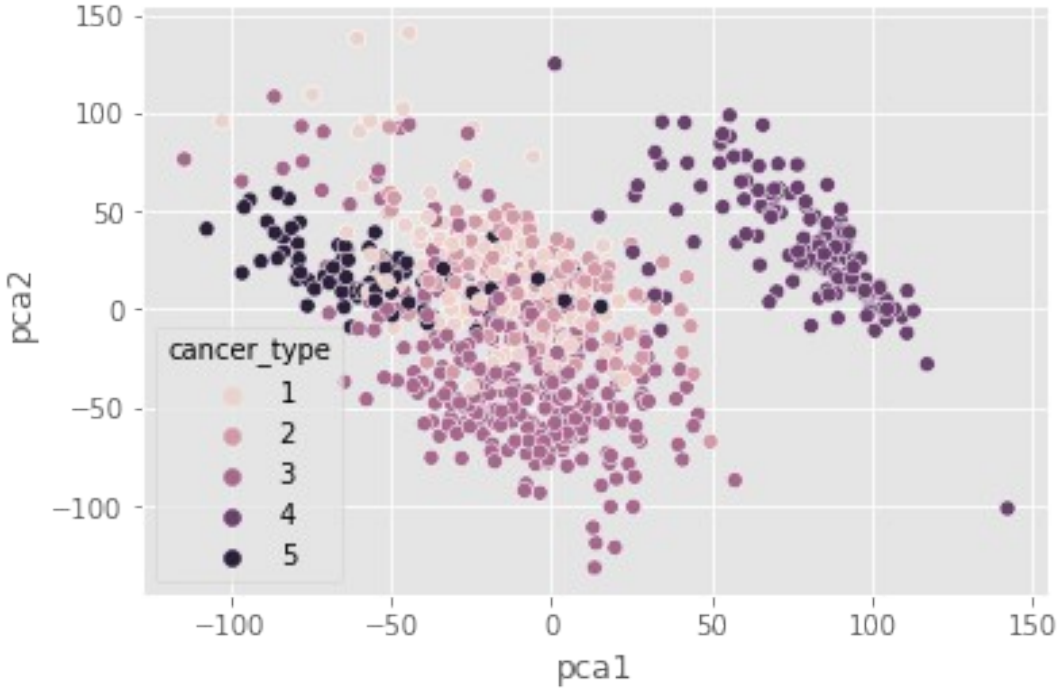
```
   gene_20525  gene_20526  gene_20527  gene_20528  gene_20529
gene_20530  \
0    9.119813   12.003135    9.650743    8.921326    5.286759
0.0
1    8.381612   12.674552   10.517059    9.397854    2.094168
0.0
2    9.911597    9.045255    9.788359   10.090470    1.683023
0.0
3    9.601208   11.392682    9.694814    9.684365    3.292001
0.0
4    9.846910   11.922439    9.217749    9.461191    5.110372
0.0

   Class
0   PRAD
1   LUAD
2   PRAD
3   PRAD
4   BRCA

[5 rows x 20533 columns]
```

```
df_cat_data['Class'] = df_cat_data['Class'].map({'PRAD': 1, 'LUAD': 2,
'BRCA': 3, 'KIRC': 4, 'COAD': 5})
df_cat_data = df_cat_data.drop(['Unnamed: 0'],axis=1)

# Put the data back on the 2 columns defined
df_pca = pd.DataFrame(X_pca_with_2)
df_pca.columns = ['pca1','pca2']

# Add the convereted categorical data for
df_pca['cancer_type']=df_cat_data['Class']
df_pca
```

```
          pca1       pca2  cancer_type
0   -57.446987  95.410981            1
1   -16.919430   0.732470            2
2   -70.345218 -19.303327            1
3   -49.161591  -9.227586            1
4   -18.132534 -51.327797            3
..         ...        ...          ...
796 -12.417385 -42.321574            3
797 -29.415554  28.526281            2
798  -4.133090  15.690014            5
799 -30.814757  33.526423            1
800 -22.344557   4.052356            1

[801 rows x 3 columns]
```

```
# Present the data on the 5 clusters using seaborn maps
sns.scatterplot(x='pca1',y='pca2', hue = 'cancer_type',data=df_pca)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7707809590>
```



PCA with n_components=.995

```
pca_with_995=PCA(.995)
X_pca_with_995 = pca_with_995.fit_transform(x_pca)
```

```
X_pca_with_995.shape
```

```
(801, 747)
```

```
X_pca_with_995
```

```
array([[-6.27554152e+01, -9.40719735e+01,  8.95198311e+01, ...,
         3.09258084e+00,  7.13597730e-01, -8.21221710e-02],
       [-2.43289636e+00,  9.05858418e+01, -1.06730787e+00, ...,
         1.39674724e-02, -3.95175744e-01, -9.49947250e-01],
       [-7.12668528e+01, -8.06460774e+00,  6.61124549e+01, ...,
         1.28898532e-01, -2.64530262e-01,  3.84594189e-01],
       ...,
       [ 1.04862615e+01,  2.15705946e+01,  4.13458784e+01, ...,
        -6.47882986e-01, -2.07256774e-01,  1.38942922e-01],
       [-5.50636049e+01, -9.23947780e+01,  8.00500394e+01, ...,
         1.74673062e+00,  2.02232239e+00, -1.92708948e+00],
       [-4.91030338e+01, -5.09976391e+01,  4.05037544e+01, ...,
         1.80367340e+00,  2.22994027e+00, -8.07255452e-01]])
```

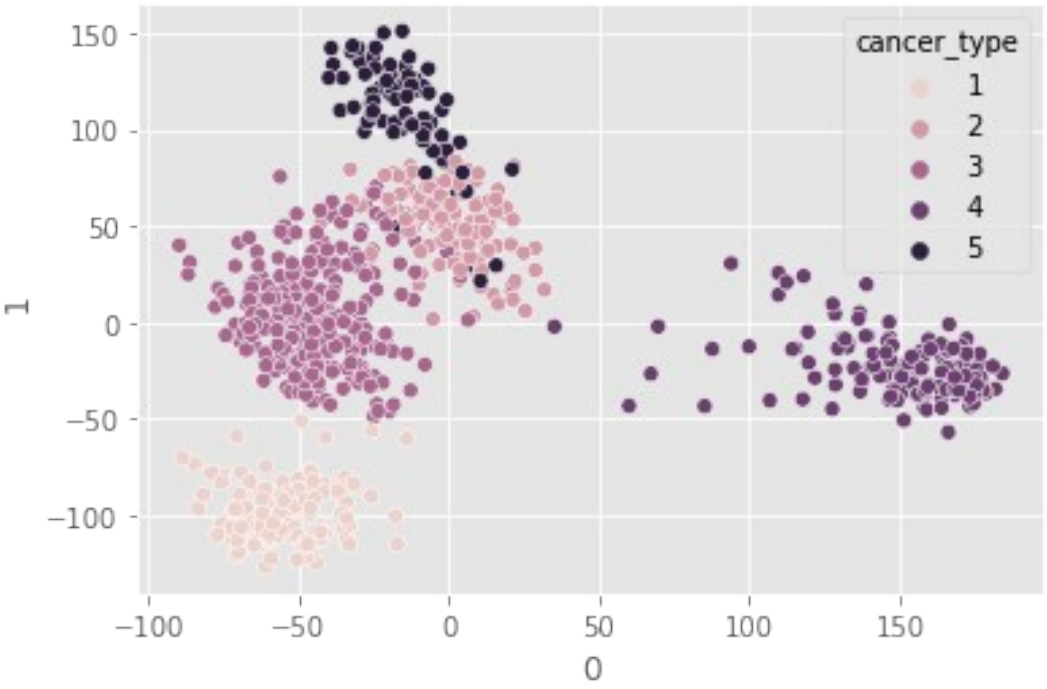
```
df_pca_995 = pd.DataFrame(X_pca_with_995)
df_pca_995['cancer_type']=df_cat_data['Class']
df_pca_995
```

```
             0          1          2          3          4           5
\
0   -62.755415 -94.071973  89.519831 -15.942567  81.423539  -13.998292

1    -2.432896  90.585842  -1.067308 -53.083120 -15.676684   60.842472

2   -71.266853  -8.064608  66.112455  81.381475  -7.525685  109.824273

3   -84.770785 -73.244566  74.181000  27.022697 -18.044895   50.116433

4   -69.560171  -9.612940 -67.497549  34.868543  -1.795849   -6.676780

..         ...        ...        ...        ...        ...         ...

796 -60.861882 -22.278633 -80.927167  42.670292   7.843763   -4.545218

797 -14.465433  53.392194  38.153904 -63.217345  22.799082   39.543441

798  10.486261  21.570595  41.345878 -59.639929  -2.163066  -96.453878

799 -55.063605 -92.394778  80.050039  -7.782015  15.180574    2.563620

800 -49.103034 -50.997639  40.503754 -31.495505 -10.361908   -1.272555


             6          7          8          9  ...       738
739  \
0     7.716073 -22.936551 -32.837892  -2.202680  ... -4.081064 -
0.626193
1    10.257369 -48.822959  14.257400 -12.214352  ...  0.215619 -
0.593678
2     5.519407 -13.364480  38.415728  -5.124731  ...  0.263786
0.328453
3    -3.495197 -11.318520   8.319656  -3.149509  ...  0.381578
0.652455
4    -2.840781  16.780157 -49.319753  10.508631  ...  1.488047
2.767486
..         ...        ...        ...        ...  ...       ...       .
..
796 -27.602910  -8.840676 -31.531870   6.380236  ... -0.780676
0.105227
797 -47.899401  39.925172 -12.413483  43.364820  ... -0.712822
0.624739
798  38.375897  46.997294  60.604643  59.967025  ...  0.269628 -
0.348648
```

```
799   8.487660  10.571657  11.710577   1.304005  ...  0.045885 -
2.222754
800   9.185948 -31.629661  40.799717  -5.265109  ... -1.429271 -
1.286569

          740       741       742       743       744       745
746  \
0   -1.265756 -0.017984 -2.740860  0.944037  3.092581  0.713598 -
0.082122
1   -0.403462  1.181537  0.490910  0.197768  0.013967 -0.395176 -
0.949947
2    0.004078  0.363928 -1.109210  0.331488  0.128899 -0.264530
0.384594
3   -3.624900 -1.203028 -2.347912  1.577992 -0.781748  0.120442 -
0.057973
4   -0.631562 -0.794275 -0.514008 -1.875969 -2.526109 -1.073803 -
1.161728
..        ...       ...       ...       ...       ...       ...
...
796 -2.001001  1.579115  0.955344  0.085881  2.667448  0.632850
0.023523
797 -0.162403 -0.238540  0.584705  1.404867  0.564251 -0.054682 -
0.905574
798 -0.531710  0.055553  0.220559  0.331122 -0.647883 -0.207257
0.138943
799 -4.115667 -0.064646 -0.447662 -0.243658  1.746731  2.022322 -
1.927089
800 -0.166544  3.095998  0.935408  2.854994  1.803673  2.229940 -
0.807255

     cancer_type
0              1
1              2
2              1
3              1
4              3
..           ...
796            3
797            2
798            5
799            1
800            1

[801 rows x 748 columns]
```

```
sns.scatterplot(x=0,y=1,hue = 'cancer_type', data=df_pca_995)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f770914f6d0>
```



How to select the number of components

```
four_master_data = master_data

five_master_data = master_data
```

```
# Define data
df4_pca = four_master_data.drop(['Unnamed: 0'], axis=1)
df4_pca = df4_pca.drop(['Class'], axis=1)
df4_pca.head()
```

```
   gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
gene_6  \
0     0.0  2.017209  3.265527  5.478487  10.431999     0.0  7.175175

1     0.0  0.592732  1.588421  7.586157   9.623011     0.0  6.816049

2     0.0  3.511759  4.327199  6.881787   9.870730     0.0  6.972130

3     0.0  3.663618  4.507649  6.659068  10.196184     0.0  7.843375

4     0.0  2.655741  2.821547  6.539454   9.738265     0.0  6.566967


     gene_7  gene_8  gene_9  ...  gene_20521  gene_20522
gene_20523  \
0  0.591871     0.0     0.0  ...    4.926711    8.210257    9.723516

1  0.000000     0.0     0.0  ...    4.593372    7.323865    9.740931
```

```
2  0.452595     0.0     0.0  ...    5.125213    8.127123   10.908640

3  0.434882     0.0     0.0  ...    6.076566    8.792959   10.141520

4  0.360982     0.0     0.0  ...    5.996032    8.891425   10.373790


   gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
0    7.220030    9.119813   12.003135    9.650743    8.921326
5.286759
1    6.256586    8.381612   12.674552   10.517059    9.397854
2.094168
2    5.401607    9.911597    9.045255    9.788359   10.090470
1.683023
3    8.942805    9.601208   11.392682    9.694814    9.684365
3.292001
4    7.181162    9.846910   11.922439    9.217749    9.461191
5.110372

   gene_20530
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0

[5 rows x 20531 columns]
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_rescaled = scaler.fit_transform(df4_pca)

# 100% of variance
from sklearn.decomposition import PCA
pca_801 = PCA(n_components = 801)
pca_801.fit(data_rescaled)
X_pca_801 = pca_801.transform(data_rescaled)

print("Variance explained by 95 components : ",
sum(pca_801.explained_variance_ratio_ * 100))
```

```
Variance explained by 95 components :  100.00000000000003
```

```
pca_801.explained_variance_ratio_ * 100
```

```
array([1.26939772e+01, 9.40227342e+00, 8.57576510e+00, 5.58780751e+00,
       4.77172956e+00, 2.78327442e+00, 2.40165349e+00, 2.25852545e+00,
       1.45690190e+00, 1.33313416e+00, 1.05892454e+00, 8.81806123e-01,
       8.51203152e-01, 7.31216746e-01, 6.42923988e-01, 5.95138322e-01,
```

5.92295108e-01, 5.56765271e-01, 4.98516572e-01, 4.84822226e-01,
4.62686834e-01, 4.36508060e-01, 4.20472556e-01, 4.09571927e-01,
3.93242673e-01, 3.85532774e-01, 3.77670869e-01, 3.60055207e-01,
3.36857943e-01, 3.29228737e-01, 3.22253680e-01, 3.11292654e-01,
3.05774329e-01, 2.91575476e-01, 2.84189640e-01, 2.77149771e-01,
2.68797419e-01, 2.67206072e-01, 2.62394709e-01, 2.49606137e-01,
2.46834687e-01, 2.45449784e-01, 2.43262686e-01, 2.39574096e-01,
2.35317421e-01, 2.31110754e-01, 2.26577781e-01, 2.19884009e-01,
2.17494494e-01, 2.15047509e-01, 2.07778997e-01, 2.04651348e-01,
2.00241205e-01, 1.99217671e-01, 1.97566987e-01, 1.94325702e-01,
1.92949786e-01, 1.89298140e-01, 1.84363416e-01, 1.83686974e-01,
1.78008496e-01, 1.75865248e-01, 1.75732435e-01, 1.74525008e-01,
1.69984021e-01, 1.65461168e-01, 1.64879967e-01, 1.62925406e-01,
1.60071073e-01, 1.57433522e-01, 1.56511040e-01, 1.53420022e-01,
1.50974001e-01, 1.50127584e-01, 1.48204122e-01, 1.47330806e-01,
1.45141142e-01, 1.42675632e-01, 1.41757437e-01, 1.40191885e-01,
1.38727506e-01, 1.36317334e-01, 1.35912345e-01, 1.33249720e-01,
1.31329553e-01, 1.30411245e-01, 1.29413522e-01, 1.26647247e-01,
1.25665404e-01, 1.25055163e-01, 1.24070735e-01, 1.22505410e-01,
1.21744855e-01, 1.20928978e-01, 1.19545864e-01, 1.18703258e-01,
1.17994490e-01, 1.16372280e-01, 1.16328642e-01, 1.15293401e-01,
1.14192619e-01, 1.13849423e-01, 1.12310448e-01, 1.11090862e-01,
1.09818053e-01, 1.08675348e-01, 1.07670645e-01, 1.07338895e-01,
1.06363704e-01, 1.05471648e-01, 1.05043957e-01, 1.03139828e-01,
1.02229708e-01, 1.00739181e-01, 1.00090757e-01, 9.96188440e-02,
9.84895448e-02, 9.81311056e-02, 9.78112828e-02, 9.68133201e-02,
9.57554205e-02, 9.50503526e-02, 9.47024378e-02, 9.43572424e-02,
9.30439464e-02, 9.23121920e-02, 9.18348466e-02, 9.15685668e-02,
9.11698630e-02, 9.06962833e-02, 9.02285838e-02, 8.95013250e-02,
8.89979070e-02, 8.80925631e-02, 8.75745623e-02, 8.72617944e-02,
8.66331442e-02, 8.62928813e-02, 8.57213946e-02, 8.49979202e-02,
8.39900853e-02, 8.34377994e-02, 8.31553498e-02, 8.24152441e-02,
8.22429530e-02, 8.18171605e-02, 8.12962113e-02, 8.08310262e-02,
8.05388545e-02, 7.99676884e-02, 7.94714035e-02, 7.90160315e-02,
7.84339478e-02, 7.82422128e-02, 7.73078568e-02, 7.71099071e-02,
7.68801119e-02, 7.61682703e-02, 7.58269194e-02, 7.54756950e-02,
7.46561632e-02, 7.43249351e-02, 7.40928973e-02, 7.38358616e-02,
7.29195341e-02, 7.27307890e-02, 7.23534333e-02, 7.22044538e-02,
7.14717022e-02, 7.12539096e-02, 7.11152366e-02, 7.03586275e-02,
7.01552049e-02, 6.97190429e-02, 6.94737960e-02, 6.89900584e-02,
6.85298906e-02, 6.81742056e-02, 6.78614406e-02, 6.76296352e-02,
6.72367712e-02, 6.65807401e-02, 6.64759414e-02, 6.60219495e-02,
6.57168481e-02, 6.55585749e-02, 6.50074924e-02, 6.48619070e-02,
6.47141098e-02, 6.45330148e-02, 6.43635907e-02, 6.37789889e-02,
6.34981021e-02, 6.33276300e-02, 6.28893864e-02, 6.26646624e-02,
6.24128572e-02, 6.20866561e-02, 6.19305224e-02, 6.17801274e-02,
6.14076914e-02, 6.09858687e-02, 6.08331064e-02, 6.06278222e-02,
6.01054626e-02, 5.99372566e-02, 5.96648627e-02, 5.93601301e-02,
5.92147384e-02, 5.89415552e-02, 5.88621110e-02, 5.85367508e-02,
5.84188880e-02, 5.82002477e-02, 5.79548666e-02, 5.76685775e-02,

5.74830445e-02, 5.70934397e-02, 5.68170124e-02, 5.66710453e-02,
5.65995772e-02, 5.63498509e-02, 5.61206409e-02, 5.59768214e-02,
5.54671598e-02, 5.52106807e-02, 5.51696921e-02, 5.50770742e-02,
5.46466460e-02, 5.45730662e-02, 5.44816748e-02, 5.39178830e-02,
5.37523559e-02, 5.36750562e-02, 5.34165467e-02, 5.33779022e-02,
5.29394744e-02, 5.27086509e-02, 5.25703307e-02, 5.22147214e-02,
5.19402968e-02, 5.18794629e-02, 5.17523134e-02, 5.14888525e-02,
5.13261503e-02, 5.12068249e-02, 5.10039821e-02, 5.08220100e-02,
5.06122011e-02, 5.04059888e-02, 5.02063888e-02, 4.99373906e-02,
4.97867073e-02, 4.96667786e-02, 4.94568150e-02, 4.93862936e-02,
4.90548519e-02, 4.88805061e-02, 4.87629503e-02, 4.87126530e-02,
4.86952489e-02, 4.84079850e-02, 4.82457810e-02, 4.81187170e-02,
4.79849140e-02, 4.78921242e-02, 4.78268091e-02, 4.77009396e-02,
4.75034703e-02, 4.73515733e-02, 4.71854233e-02, 4.70268602e-02,
4.68121355e-02, 4.67428370e-02, 4.65949411e-02, 4.63183604e-02,
4.62897601e-02, 4.61389241e-02, 4.60205246e-02, 4.58838513e-02,
4.55096302e-02, 4.54358331e-02, 4.53073975e-02, 4.52596074e-02,
4.50349940e-02, 4.47823670e-02, 4.47463582e-02, 4.46293385e-02,
4.45244462e-02, 4.44457224e-02, 4.43398145e-02, 4.40059812e-02,
4.39662004e-02, 4.37979714e-02, 4.36055443e-02, 4.35607894e-02,
4.34334409e-02, 4.33418805e-02, 4.31637250e-02, 4.30791326e-02,
4.29877606e-02, 4.27780530e-02, 4.25892255e-02, 4.25344648e-02,
4.22436222e-02, 4.21458602e-02, 4.20183503e-02, 4.18808000e-02,
4.17439188e-02, 4.16299230e-02, 4.15692650e-02, 4.15114251e-02,
4.13923169e-02, 4.11264611e-02, 4.10383815e-02, 4.09512585e-02,
4.08669758e-02, 4.07443126e-02, 4.06857690e-02, 4.05042727e-02,
4.03518898e-02, 4.02949075e-02, 4.01705373e-02, 3.99824067e-02,
3.99198534e-02, 3.96902004e-02, 3.96313948e-02, 3.95336057e-02,
3.94528789e-02, 3.93248217e-02, 3.92708909e-02, 3.91867670e-02,
3.90021447e-02, 3.88807706e-02, 3.87427578e-02, 3.87184244e-02,
3.86414225e-02, 3.85300422e-02, 3.83677647e-02, 3.82604529e-02,
3.80739166e-02, 3.80303342e-02, 3.78513739e-02, 3.78287295e-02,
3.77124199e-02, 3.75563013e-02, 3.75273717e-02, 3.72752578e-02,
3.71846187e-02, 3.70705844e-02, 3.70167437e-02, 3.69115272e-02,
3.68019080e-02, 3.67586513e-02, 3.65385933e-02, 3.64509389e-02,
3.63438928e-02, 3.61931998e-02, 3.61301068e-02, 3.59522653e-02,
3.58931326e-02, 3.58309158e-02, 3.58031987e-02, 3.57062253e-02,
3.56925195e-02, 3.55015211e-02, 3.54704146e-02, 3.53557671e-02,
3.52000413e-02, 3.51173755e-02, 3.50381087e-02, 3.50167367e-02,
3.48959464e-02, 3.48047856e-02, 3.47398849e-02, 3.45832682e-02,
3.45400060e-02, 3.44662243e-02, 3.43632488e-02, 3.42468905e-02,
3.41036296e-02, 3.40763439e-02, 3.39151500e-02, 3.38129251e-02,
3.37495548e-02, 3.36444387e-02, 3.35771763e-02, 3.34754796e-02,
3.34017820e-02, 3.32774187e-02, 3.32749910e-02, 3.31253312e-02,
3.30667941e-02, 3.29360977e-02, 3.28969271e-02, 3.27767721e-02,
3.27260098e-02, 3.26427422e-02, 3.25608212e-02, 3.24794391e-02,
3.24110145e-02, 3.23685660e-02, 3.22692339e-02, 3.21210363e-02,
3.20587347e-02, 3.19535901e-02, 3.18860607e-02, 3.17834782e-02,
3.17581103e-02, 3.16441193e-02, 3.15355401e-02, 3.14834652e-02,
3.14027104e-02, 3.12675117e-02, 3.11732183e-02, 3.10922182e-02,

3.10097777e-02, 3.09888425e-02, 3.09399978e-02, 3.07533940e-02,
3.06800693e-02, 3.06484929e-02, 3.05969866e-02, 3.04885013e-02,
3.03794719e-02, 3.03109934e-02, 3.01787867e-02, 3.01550493e-02,
3.01008176e-02, 2.99914527e-02, 2.99100626e-02, 2.98285449e-02,
2.98109877e-02, 2.97466314e-02, 2.96504305e-02, 2.95373339e-02,
2.94821948e-02, 2.93933246e-02, 2.93473318e-02, 2.93295473e-02,
2.92536594e-02, 2.91719351e-02, 2.90206820e-02, 2.90049095e-02,
2.89838845e-02, 2.88932657e-02, 2.88256023e-02, 2.87045554e-02,
2.85969946e-02, 2.85735264e-02, 2.84906888e-02, 2.84037874e-02,
2.83835326e-02, 2.82730659e-02, 2.82027161e-02, 2.81829245e-02,
2.80792505e-02, 2.79858174e-02, 2.79709049e-02, 2.79424842e-02,
2.78705411e-02, 2.78077217e-02, 2.77460115e-02, 2.76417294e-02,
2.75146095e-02, 2.74888127e-02, 2.74489215e-02, 2.73512306e-02,
2.73321949e-02, 2.72488268e-02, 2.71271851e-02, 2.70848200e-02,
2.69957391e-02, 2.69843788e-02, 2.68111371e-02, 2.67973351e-02,
2.67640546e-02, 2.66923861e-02, 2.65589704e-02, 2.65399792e-02,
2.65019444e-02, 2.64241138e-02, 2.63803286e-02, 2.62963589e-02,
2.62461100e-02, 2.62048995e-02, 2.60999185e-02, 2.60411163e-02,
2.59516180e-02, 2.58988935e-02, 2.58478569e-02, 2.57470336e-02,
2.57040945e-02, 2.56727655e-02, 2.56244901e-02, 2.54992043e-02,
2.54184601e-02, 2.53728993e-02, 2.53403760e-02, 2.52417751e-02,
2.52339714e-02, 2.52068203e-02, 2.51557608e-02, 2.50652080e-02,
2.50283052e-02, 2.49492954e-02, 2.49130773e-02, 2.48664138e-02,
2.47783509e-02, 2.47709453e-02, 2.46391415e-02, 2.45941234e-02,
2.45299213e-02, 2.44037106e-02, 2.43701300e-02, 2.42763861e-02,
2.42414959e-02, 2.41970803e-02, 2.41482555e-02, 2.41146317e-02,
2.39827413e-02, 2.39423976e-02, 2.38906295e-02, 2.38502502e-02,
2.37349782e-02, 2.36789049e-02, 2.36082188e-02, 2.35727557e-02,
2.35348468e-02, 2.34881942e-02, 2.34577916e-02, 2.34167665e-02,
2.32924201e-02, 2.32658117e-02, 2.32313882e-02, 2.31716173e-02,
2.31382266e-02, 2.30353963e-02, 2.29884096e-02, 2.29530842e-02,
2.28926414e-02, 2.28466905e-02, 2.28328493e-02, 2.27469113e-02,
2.26435607e-02, 2.26039686e-02, 2.25565382e-02, 2.25348191e-02,
2.24762801e-02, 2.23732946e-02, 2.23662529e-02, 2.23257728e-02,
2.22570615e-02, 2.21824739e-02, 2.21387345e-02, 2.20850154e-02,
2.20656296e-02, 2.20294763e-02, 2.19315991e-02, 2.19103746e-02,
2.18301328e-02, 2.18100171e-02, 2.17745972e-02, 2.16022710e-02,
2.15628794e-02, 2.15329380e-02, 2.14958735e-02, 2.14447850e-02,
2.13420825e-02, 2.13293314e-02, 2.12650990e-02, 2.12302929e-02,
2.12215312e-02, 2.11383969e-02, 2.10526929e-02, 2.10262273e-02,
2.09934749e-02, 2.09413212e-02, 2.08957603e-02, 2.08786964e-02,
2.08257332e-02, 2.07108407e-02, 2.06898642e-02, 2.05867909e-02,
2.05642375e-02, 2.05212678e-02, 2.04636061e-02, 2.04165308e-02,
2.03762699e-02, 2.03467609e-02, 2.03114772e-02, 2.02615672e-02,
2.01798483e-02, 2.01331044e-02, 2.00782319e-02, 2.00576373e-02,
2.00411393e-02, 1.99360289e-02, 1.98617391e-02, 1.98310710e-02,
1.97740182e-02, 1.96910978e-02, 1.96579620e-02, 1.96313517e-02,
1.95652369e-02, 1.95276583e-02, 1.95016156e-02, 1.94716336e-02,
1.94189802e-02, 1.93571151e-02, 1.92953047e-02, 1.92633157e-02,
1.91762790e-02, 1.91228885e-02, 1.90984045e-02, 1.90344282e-02,

```
       1.89916448e-02, 1.89562541e-02, 1.89091121e-02, 1.88689235e-02,
       1.88145677e-02, 1.87779089e-02, 1.87123548e-02, 1.86527017e-02,
       1.86015376e-02, 1.85537110e-02, 1.85197993e-02, 1.84663447e-02,
       1.84484277e-02, 1.83515206e-02, 1.83224187e-02, 1.82824924e-02,
       1.82445333e-02, 1.81868802e-02, 1.81154330e-02, 1.80935452e-02,
       1.80383518e-02, 1.79606543e-02, 1.79487896e-02, 1.79103110e-02,
       1.78829119e-02, 1.78033161e-02, 1.77341332e-02, 1.77099980e-02,
       1.76538904e-02, 1.75667449e-02, 1.75383016e-02, 1.74997622e-02,
       1.73955525e-02, 1.73770186e-02, 1.73655515e-02, 1.73090992e-02,
       1.72227221e-02, 1.71769252e-02, 1.71474257e-02, 1.71057728e-02,
       1.71030471e-02, 1.70005464e-02, 1.69483383e-02, 1.69301961e-02,
       1.68966532e-02, 1.68577851e-02, 1.68338597e-02, 1.68114867e-02,
       1.67127707e-02, 1.66961519e-02, 1.66388225e-02, 1.66294397e-02,
       1.65352290e-02, 1.65257779e-02, 1.64551227e-02, 1.64057524e-02,
       1.63822836e-02, 1.63482768e-02, 1.62951750e-02, 1.62381678e-02,
       1.62208416e-02, 1.61513917e-02, 1.61077016e-02, 1.60618723e-02,
       1.60130404e-02, 1.59759385e-02, 1.59678663e-02, 1.58451303e-02,
       1.58226594e-02, 1.57936898e-02, 1.57618814e-02, 1.57145263e-02,
       1.56532297e-02, 1.56071924e-02, 1.55380261e-02, 1.54989844e-02,
       1.54853133e-02, 1.54200699e-02, 1.53867789e-02, 1.53356277e-02,
       1.52958902e-02, 1.52560079e-02, 1.52213630e-02, 1.51784010e-02,
       1.51233215e-02, 1.50842914e-02, 1.50761297e-02, 1.49599053e-02,
       1.49099978e-02, 1.48862468e-02, 1.48566249e-02, 1.48460777e-02,
       1.47286460e-02, 1.47116910e-02, 1.46533132e-02, 1.46274451e-02,
       1.46123566e-02, 1.45287817e-02, 1.45013149e-02, 1.44675245e-02,
       1.44144148e-02, 1.44031809e-02, 1.43599970e-02, 1.43224270e-02,
       1.42692552e-02, 1.42247712e-02, 1.41773926e-02, 1.40909225e-02,
       1.40350561e-02, 1.39728883e-02, 1.39236638e-02, 1.38739053e-02,
       1.38044879e-02, 1.37819796e-02, 1.37437858e-02, 1.36627455e-02,
       1.36347448e-02, 1.35877430e-02, 1.35382359e-02, 1.34891693e-02,
       1.34750982e-02, 1.33522172e-02, 1.33092167e-02, 1.32746389e-02,
       1.32295823e-02, 1.31878842e-02, 1.31407484e-02, 1.30685710e-02,
       1.30267969e-02, 1.30149340e-02, 1.29557414e-02, 1.28898244e-02,
       1.28647879e-02, 1.28207043e-02, 1.27702951e-02, 1.27481785e-02,
       1.26271443e-02, 1.26038750e-02, 1.25662142e-02, 1.25102179e-02,
       1.24638116e-02, 1.23345727e-02, 1.22751465e-02, 1.22092831e-02,
       1.22070811e-02, 1.21924023e-02, 1.21040463e-02, 1.20920681e-02,
       1.20402216e-02, 1.19984863e-02, 1.19793197e-02, 1.18550687e-02,
       1.17772095e-02, 1.17617010e-02, 1.16248396e-02, 1.15983088e-02,
       1.15140838e-02, 1.14429032e-02, 1.14245635e-02, 1.13652645e-02,
       1.13605403e-02, 1.11821381e-02, 1.11349098e-02, 1.10216626e-02,
       1.09981664e-02, 1.09267733e-02, 1.08562289e-02, 1.07631743e-02,
       1.06566264e-02, 1.06282472e-02, 1.05676552e-02, 1.04406022e-02,
       1.02847900e-02, 1.02214444e-02, 1.01701477e-02, 1.00823950e-02,
       9.95937655e-03, 9.82753898e-03, 9.67114298e-03, 9.64046154e-03,
       9.37541667e-03, 9.25473780e-03, 8.98641634e-03, 8.92022032e-03,
       6.43988351e-30])
```

```
np.cumsum(pca_801.explained_variance_ratio_ * 100)
```

```
array([ 12.6939772 ,  22.09625062,  30.67201572,  36.25982323,
        41.0315528 ,  43.81482722,  46.21648071,  48.47500616,
        49.93190806,  51.26504222,  52.32396676,  53.20577288,
        54.05697604,  54.78819278,  55.43111677,  56.02625509,
        56.6185502 ,  57.17531547,  57.67383204,  58.15865427,
        58.6213411 ,  59.05784916,  59.47832172,  59.88789365,
        60.28113632,  60.66666909,  61.04433996,  61.40439517,
        61.74125311,  62.07048185,  62.39273553,  62.70402818,
        63.00980251,  63.30137799,  63.58556763,  63.8627174 ,
        64.13151482,  64.39872089,  64.6611156 ,  64.91072174,
        65.15755642,  65.40300621,  65.64626889,  65.88584299,
        66.12116041,  66.35227117,  66.57884895,  66.79873296,
        67.01622745,  67.23127496,  67.43905396,  67.6437053 ,
        67.84394651,  68.04316418,  68.24073117,  68.43505687,
        68.62800665,  68.81730479,  69.00166821,  69.18535519,
        69.36336368,  69.53922893,  69.71496136,  69.88948637,
        70.05947039,  70.22493156,  70.38981153,  70.55273693,
        70.71280801,  70.87024153,  71.02675257,  71.18017259,
        71.33114659,  71.48127418,  71.6294783 ,  71.7768091 ,
        71.92195025,  72.06462588,  72.20638332,  72.3465752 ,
        72.48530271,  72.62162004,  72.75753239,  72.89078211,
        73.02211166,  73.15252291,  73.28193643,  73.40858367,
        73.53424908,  73.65930424,  73.78337498,  73.90588039,
        74.02762524,  74.14855422,  74.26810008,  74.38680334,
        74.50479783,  74.62117011,  74.73749875,  74.85279215,
        74.96698477,  75.0808342 ,  75.19314464,  75.30423551,
        75.41405356,  75.52272891,  75.63039955,  75.73773845,
        75.84410215,  75.9495738 ,  76.05461776,  76.15775758,
        76.25998729,  76.36072647,  76.46081723,  76.56043607,
        76.65892562,  76.75705673,  76.85486801,  76.95168133,
        77.04743675,  77.1424871 ,  77.23718954,  77.33154678,
        77.42459073,  77.51690292,  77.60873777,  77.70030633,
        77.7914762 ,  77.88217248,  77.97240106,  78.06190239,
        78.1509003 ,  78.23899286,  78.32656742,  78.41382922,
        78.50046236,  78.58675524,  78.67247664,  78.75747456,
        78.84146464,  78.92490244,  79.00805779,  79.09047303,
        79.17271599,  79.25453315,  79.33582936,  79.41666039,
        79.49719924,  79.57716693,  79.65663833,  79.73565436,
        79.81408831,  79.89233052,  79.96963838,  80.04674829,
        80.1236284 ,  80.19979667,  80.27562359,  80.35109928,
        80.42575545,  80.50008038,  80.57417328,  80.64800914,
        80.72092868,  80.79365946,  80.8660129 ,  80.93821735,
        81.00968905,  81.08094296,  81.1520582 ,  81.22241683,
        81.29257203,  81.36229108,  81.43176487,  81.50075493,
        81.56928482,  81.63745903,  81.70532047,  81.7729501 ,
        81.84018687,  81.90676761,  81.97324355,  82.0392655 ,
        82.10498235,  82.17054093,  82.23554842,  82.30041033,
        82.36512444,  82.42965745,  82.49402104,  82.55780003,
        82.62129813,  82.68462576,  82.74751515,  82.81017981,
        82.87259267,  82.93467932,  82.99660985,  83.05838997,
```

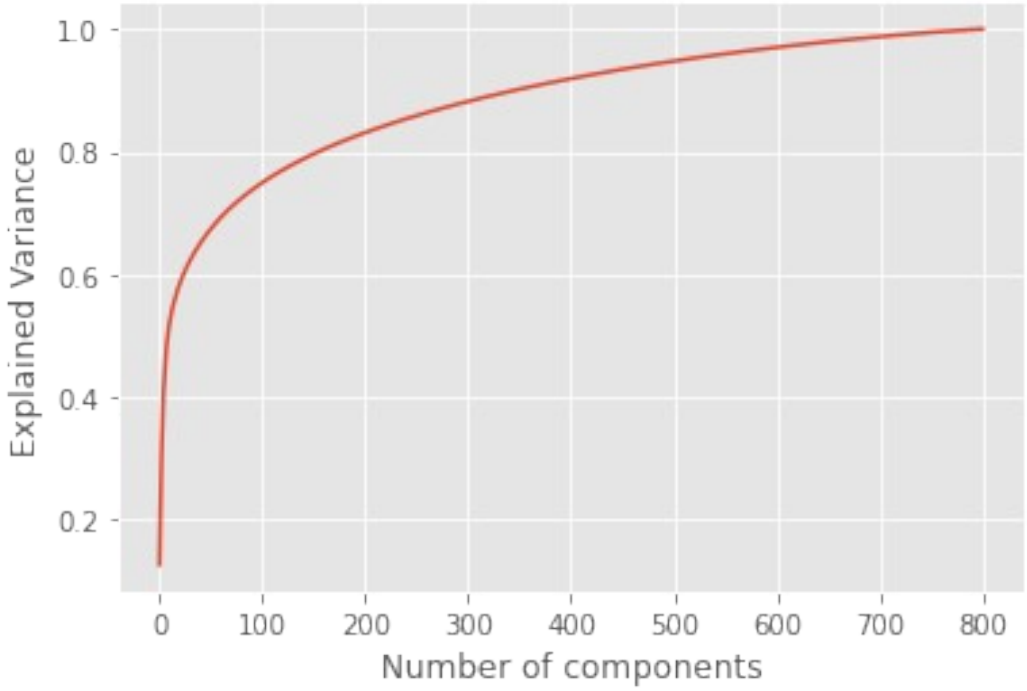
83.11979767, 83.18078353, 83.24161664, 83.30224446,
83.36234993, 83.42228718, 83.48195205, 83.54131218,
83.60052691, 83.65946847, 83.71833058, 83.77686733,
83.83528622, 83.89348647, 83.95144133, 84.00910991,
84.06659295, 84.12368639, 84.18050341, 84.23717445,
84.29377403, 84.35012388, 84.40624452, 84.46222134,
84.5176885 , 84.57289918, 84.62806888, 84.68314595,
84.7377926 , 84.79236566, 84.84684734, 84.90076522,
84.95451758, 85.00819263, 85.06160918, 85.11498708,
85.16792655, 85.22063521, 85.27320554, 85.32542026,
85.37736055, 85.42924002, 85.48099233, 85.53248118,
85.58380733, 85.63501416, 85.68601814, 85.73684015,
85.78745235, 85.83785834, 85.88806473, 85.93800212,
85.98778883, 86.03745561, 86.08691242, 86.13629871,
86.18535357, 86.23423407, 86.28299702, 86.33170968,
86.38040492, 86.42881291, 86.47705869, 86.52517741,
86.57316232, 86.62105445, 86.66888126, 86.71658219,
86.76408567, 86.81143724, 86.85862266, 86.90564952,
86.95246166, 86.99920449, 87.04579944, 87.0921178 ,
87.13840756, 87.18454648, 87.230567 , 87.27645086,
87.32196049, 87.36739632, 87.41270372, 87.45796332,
87.50299832, 87.54778069, 87.59252704, 87.63715638,
87.68168083, 87.72612655, 87.77046637, 87.81447235,
87.85843855, 87.90223652, 87.94584206, 87.98940285,
88.03283629, 88.07617817, 88.1193419 , 88.16242103,
88.20540879, 88.24818684, 88.29077607, 88.33331053,
88.37555416, 88.41770002, 88.45971837, 88.50159917,
88.54334309, 88.58497301, 88.62654227, 88.6680537 ,
88.70944602, 88.75057248, 88.79161086, 88.83256212,
88.87342909, 88.91417341, 88.95485917, 88.99536345,
89.03571534, 89.07601024, 89.11618078, 89.15616319,
89.19608304, 89.23577324, 89.27540464, 89.31493824,
89.35439112, 89.39371594, 89.43298683, 89.4721736 ,
89.51117575, 89.55005652, 89.58879927, 89.6275177 ,
89.66615912, 89.70468916, 89.74305693, 89.78131738,
89.8193913 , 89.85742163, 89.89527301, 89.93310174,
89.97081416, 90.00837046, 90.04589783, 90.08317309,
90.1203577 , 90.15742829, 90.19444503, 90.23135656,
90.26815847, 90.30491712, 90.34145571, 90.37790665,
90.41425054, 90.45044374, 90.48657385, 90.52252612,
90.55841925, 90.59425016, 90.63005336, 90.66575959,
90.70145211, 90.73695363, 90.77242404, 90.80777981,
90.84297985, 90.87809723, 90.91313534, 90.94815207,
90.98304802, 91.01785281, 91.05259269, 91.08717596,
91.12171596, 91.15618219, 91.19054544, 91.22479233,
91.25889596, 91.2929723 , 91.32688745, 91.36070038,
91.39444993, 91.42809437, 91.46167155, 91.49514703,
91.52854881, 91.56182623, 91.59510122, 91.62822655,
91.66129334, 91.69422944, 91.72712637, 91.75990314,
91.79262915, 91.82527189, 91.85783271, 91.89031215,

91.92272317, 91.95509173, 91.98736097, 92.019482 ,
92.05154074, 92.08349433, 92.11538039, 92.14716387,
92.17892198, 92.2105661 , 92.24210164, 92.2735851 ,
92.30498781, 92.33625532, 92.36742854, 92.39852076,
92.42953054, 92.46051938, 92.49145938, 92.52221277,
92.55289284, 92.58354133, 92.61413832, 92.64462682,
92.67500629, 92.70531729, 92.73549607, 92.76565112,
92.79575194, 92.82574339, 92.85565346, 92.885482 ,
92.91529299, 92.94503962, 92.97469005, 93.00422739,
93.03370958, 93.0631029 , 93.09245024, 93.12177978,
93.15103344, 93.18020538, 93.20922606, 93.23823097,
93.26721485, 93.29610812, 93.32493372, 93.35363828,
93.38223527, 93.4108088 , 93.43929949, 93.46770327,
93.49608681, 93.52435987, 93.55256259, 93.58074551,
93.60882476, 93.63681058, 93.66478149, 93.69272397,
93.72059451, 93.74840223, 93.77614825, 93.80378997,
93.83130458, 93.8587934 , 93.88624232, 93.91359355,
93.94092574, 93.96817457, 93.99530176, 94.02238658,
94.04938231, 94.07636669, 94.10317783, 94.12997517,
94.15673922, 94.18343161, 94.20999058, 94.23653056,
94.2630325 , 94.28945661, 94.31583694, 94.3421333 ,
94.36837941, 94.39458431, 94.42068423, 94.44672535,
94.47267696, 94.49857586, 94.52442371, 94.55017075,
94.57587484, 94.60154761, 94.6271721 , 94.6526713 ,
94.67808976, 94.70346266, 94.72880304, 94.75404481,
94.77927878, 94.8044856 , 94.82964137, 94.85470657,
94.87973488, 94.90468417, 94.92959725, 94.95446367,
94.97924202, 95.00401296, 95.0286521 , 95.05324623,
95.07777615, 95.10217986, 95.12654999, 95.15082637,
95.17506787, 95.19926495, 95.22341321, 95.24752784,
95.27151058, 95.29545298, 95.31934361, 95.34319386,
95.36692883, 95.39060774, 95.41421596, 95.43778871,
95.46132356, 95.48481175, 95.50826955, 95.53168631,
95.55497873, 95.57824454, 95.60147593, 95.62464755,
95.64778578, 95.67082117, 95.69380958, 95.71676267,
95.73965531, 95.762502 , 95.78533485, 95.80808176,
95.83072532, 95.85332929, 95.87588583, 95.89842065,
95.92089693, 95.94327022, 95.96563647, 95.98796225,
96.01021931, 96.03240178, 96.05454052, 96.07662553,
96.09869116, 96.12072064, 96.14265224, 96.16456261,
96.18639274, 96.20820276, 96.22997736, 96.25157963,
96.27314251, 96.29467545, 96.31617132, 96.33761611,
96.35895819, 96.38028752, 96.40155262, 96.42278291,
96.44400444, 96.46514284, 96.48619553, 96.50722176,
96.52821523, 96.54915656, 96.57005232, 96.59093101,
96.61175675, 96.63246759, 96.65315745, 96.67374424,
96.69430848, 96.71482975, 96.73529335, 96.75570988,
96.77608615, 96.79643291, 96.81674439, 96.83700596,
96.85718581, 96.87731891, 96.89739714, 96.91745478,
96.93749592, 96.95743195, 96.97729369, 96.99712476,

97.01689878, 97.03658988, 97.05624784, 97.07587919,
97.09544443, 97.11497208, 97.1344737 , 97.15394533,
97.17336431, 97.19272143, 97.21201673, 97.23128005,
97.25045633, 97.26957922, 97.28867762, 97.30771205,
97.32670369, 97.34565995, 97.36456906, 97.38343798,
97.40225255, 97.42103046, 97.43974282, 97.45839552,
97.47699705, 97.49555077, 97.51407056, 97.53253691,
97.55098534, 97.56933686, 97.58765928, 97.60594177,
97.6241863 , 97.64237318, 97.66048862, 97.67858216,
97.69662051, 97.71458117, 97.73252996, 97.75044027,
97.76832318, 97.7861265 , 97.80386063, 97.82157063,
97.83922452, 97.85679126, 97.87432956, 97.89182933,
97.90922488, 97.9266019 , 97.94396745, 97.96127655,
97.97849927, 97.99567619, 98.01282362, 98.02992939,
98.04703244, 98.06403299, 98.08098132, 98.09791152,
98.11480817, 98.13166596, 98.14849982, 98.16531131,
98.18202408, 98.19872023, 98.21535905, 98.23198849,
98.24852372, 98.2650495 , 98.28150462, 98.29791037,
98.31429266, 98.33064093, 98.34693611, 98.36317428,
98.37939512, 98.39554651, 98.41165421, 98.42771608,
98.44372912, 98.45970506, 98.47567293, 98.49151806,
98.50734072, 98.52313441, 98.53889629, 98.55461082,
98.57026405, 98.58587124, 98.60140926, 98.61690825,
98.63239356, 98.64781363, 98.66320041, 98.67853604,
98.69383193, 98.70908794, 98.7243093 , 98.7394877 ,
98.75461102, 98.76969531, 98.78477144, 98.79973135,
98.81464135, 98.82952759, 98.84438422, 98.85923029,
98.87395894, 98.88867063, 98.90332395, 98.91795139,
98.93256375, 98.94709253, 98.96159384, 98.97606137,
98.99047578, 99.00487896, 99.01923896, 99.03356139,
99.04783064, 99.06205541, 99.07623281, 99.09032373,
99.10435879, 99.11833167, 99.13225534, 99.14612924,
99.15993373, 99.17371571, 99.1874595 , 99.20112224,
99.21475699, 99.22834473, 99.24188297, 99.25537213,
99.26884723, 99.28219945, 99.29550867, 99.30878331,
99.32201289, 99.33520077, 99.34834152, 99.36141009,
99.37443689, 99.38745182, 99.40040756, 99.41329739,
99.42616218, 99.43898288, 99.45175318, 99.46450135,
99.4771285 , 99.48973237, 99.50229859, 99.51480881,
99.52727262, 99.53960719, 99.55188234, 99.56409162,
99.5762987 , 99.5884911 , 99.60059515, 99.61268722,
99.62472744, 99.63672592, 99.64870524, 99.66056031,
99.67233752, 99.68409922, 99.69572406, 99.70732237,
99.71883646, 99.73027936, 99.74170392, 99.75306919,
99.76442973, 99.77561187, 99.78674678, 99.79776844,
99.8087666 , 99.81969338, 99.83054961, 99.84131278,
99.85196941, 99.86259765, 99.87316531, 99.88360591,
99.8938907 , 99.90411215, 99.91428229, 99.92436469,
99.93432407, 99.9441516 , 99.95382275, 99.96346321,

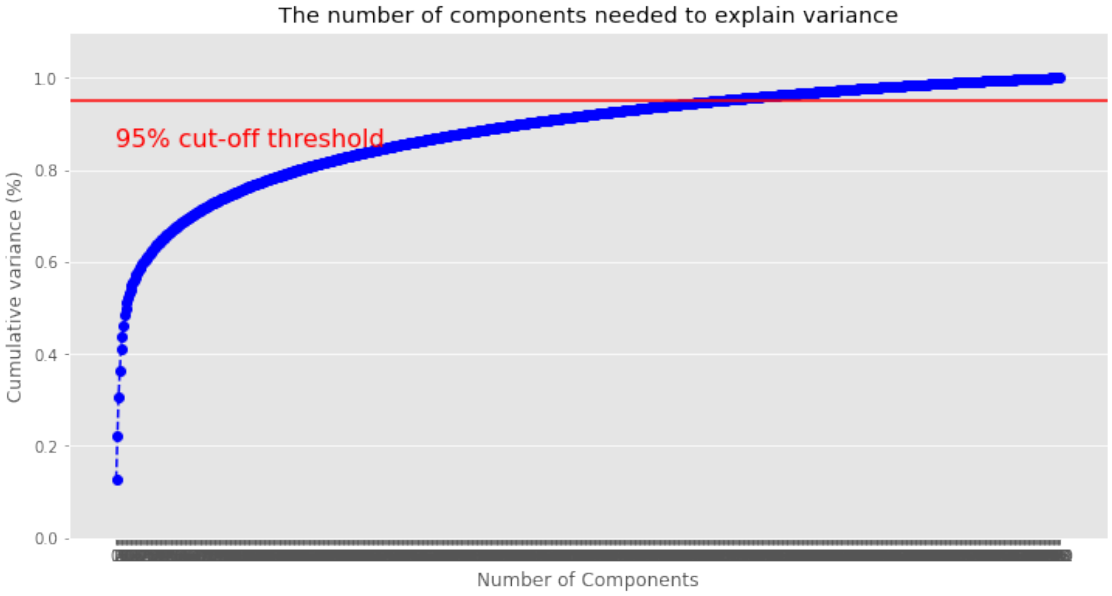
```
        99.97283863,  99.98209336,  99.99107978, 100.        ,
       100.        ])
```

```
plt.plot(np.cumsum(pca_801.explained_variance_ratio_))
plt.xlabel("Number of components")
plt.ylabel("Explained Variance")
plt.savefig("elbow_plot.png", dpi=100)
```



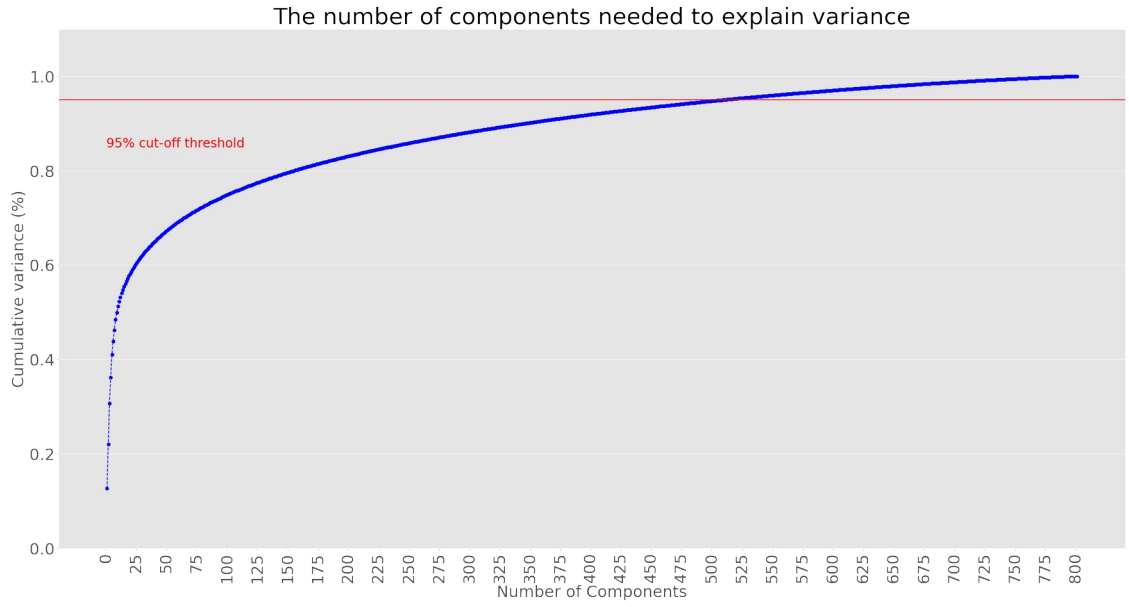

```
pca_801_fit = PCA().fit(data_rescaled)

# % matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (12,6)

fig, ax = plt.subplots()
xi = np.arange(1, 802, step=1)
y = np.cumsum(pca_801_fit.explained_variance_ratio_)

plt.ylim(0.0,1.1)
plt.plot(xi, y, marker='o', linestyle='--', color='b')

plt.xlabel('Number of Components')
plt.xticks(np.arange(0, 802, step=1)) #change from 0-based array index
to 1-based human-readable label
plt.ylabel('Cumulative variance (%)')
plt.title('The number of components needed to explain variance')

plt.axhline(y=0.95, color='r', linestyle='-')
```

```
plt.text(0.5, 0.85, '95% cut-off threshold', color = 'red',
fontsize=16)

ax.grid(axis='x')
plt.show()
```




```
# % matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (36,18)  # (56, 36)

# creating a dictionary
font = {'size': 30}

# using rc function
plt.rc('font', **font)

fig, ax = plt.subplots()
xi = np.arange(1, 802, step=1)
y = np.cumsum(pca_801_fit.explained_variance_ratio_)

plt.ylim(0.0,1.1)
plt.plot(xi, y, marker='o', linestyle='--', color='b')

plt.xlabel('Number of Components', fontsize=30)

plt.xticks(np.arange(0, 802, step=25), rotation=90) #change from 0-
based array index to 1-based human-readable label


plt.ylabel('Cumulative variance (%)', fontsize=30)
```

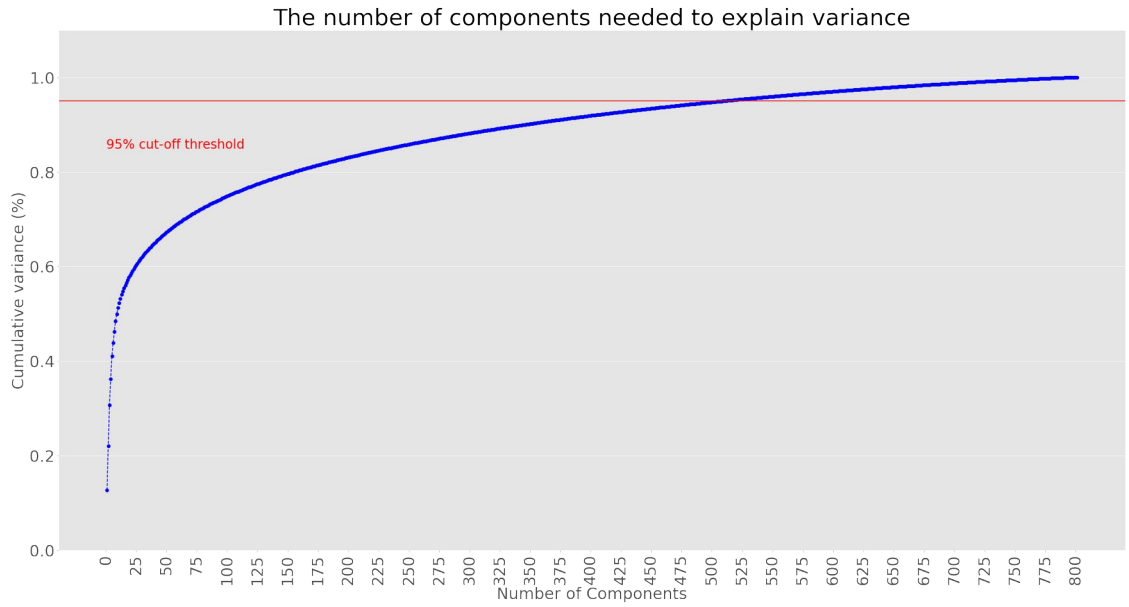
```
plt.title('The number of components needed to explain variance')

# plt.setp(xi.get_xticklabels(), rotation=30,
horizontalalignment='right')

plt.axhline(y=0.95, color='r', linestyle='-')
plt.text(0.5, 0.85, '95% cut-off threshold', color = 'red',
fontsize=24)

ax.grid(axis='x')
plt.show()
plt.savefig("clear_elbow_plot.png")
```




```
<Figure size 2592x1296 with 0 Axes>
```

```
# % matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (36,18)  # (56, 36)

# creating a dictionary
font = {'size': 30}

# using rc function
plt.rc('font', **font)

fig, ax = plt.subplots()
xi = np.arange(1, 802, step=1)
y = np.cumsum(pca_801_fit.explained_variance_ratio_)

plt.ylim(0.0,1.1)
plt.plot(xi, y, marker='o', linestyle='--', color='b')
```

```
plt.xlabel('Number of Components', fontsize=30)

plt.xticks(np.arange(0, 802, step=25), rotation=90) #change from 0-
based array index to 1-based human-readable label


plt.ylabel('Cumulative variance (%)', fontsize=30)
plt.title('The number of components needed to explain variance')

# plt.setp(xi.get_xticklabels(), rotation=30,
horizontalalignment='right')

plt.axhline(y=0.95, color='r', linestyle='-')
plt.text(0.5, 0.85, '95% cut-off threshold', color = 'red',
fontsize=24)

ax.grid(axis='x')
plt.show()
plt.savefig("clear_elbow_plot.png", dpi=100)
```




```
<Figure size 2592x1296 with 0 Axes>
```

Dimensionality reduction using TSNE

Reference - https://www.youtube.com/watch?v=NEaUSP4YerM&t=625s

```
df_tsne_data = master_data
non_numeric = ['Unnamed: 0','Class']
df_tsne_data = df_tsne_data.drop(non_numeric, axis=1)
df_tsne_data
```

	gene_0	gene_1	gene_2	gene_3	gene_4	gene_5	gene_6
0	0.0	2.017209	3.265527	5.478487	10.431999	0.0	7.175175
1	0.0	0.592732	1.588421	7.586157	9.623011	0.0	6.816049
2	0.0	3.511759	4.327199	6.881787	9.870730	0.0	6.972130
3	0.0	3.663618	4.507649	6.659068	10.196184	0.0	7.843375
4	0.0	2.655741	2.821547	6.539454	9.738265	0.0	6.566967
..	...	...	...	...	...	...	...
796	0.0	1.865642	2.718197	7.350099	10.006003	0.0	6.764792
797	0.0	3.942955	4.453807	6.346597	10.056868	0.0	7.320331
798	0.0	3.249582	3.707492	8.185901	9.504082	0.0	7.536589
799	0.0	2.590339	2.787976	7.318624	9.987136	0.0	9.213464
800	0.0	2.325242	3.805932	6.530246	9.560367	0.0	7.957027

	gene_7	gene_8	gene_9	...	gene_20521	gene_20522	gene_20523
0	0.591871	0.0	0.0	...	4.926711	8.210257	9.723516
1	0.000000	0.0	0.0	...	4.593372	7.323865	9.740931
2	0.452595	0.0	0.0	...	5.125213	8.127123	10.908640
3	0.434882	0.0	0.0	...	6.076566	8.792959	10.141520
4	0.360982	0.0	0.0	...	5.996032	8.891425	10.373790
..	...	...	...	...	...	...	...
796	0.496922	0.0	0.0	...	6.088133	9.118313	10.004852
797	0.000000	0.0	0.0	...	6.371876	9.623335	9.823921
798	1.811101	0.0	0.0	...	5.719386	8.610704	10.485517
799	0.000000	0.0	0.0	...	5.785237	8.605387	11.004677
800	0.000000	0.0	0.0	...	6.403075	8.594354	10.243079

```
     gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
0      7.220030    9.119813   12.003135    9.650743    8.921326
5.286759
1      6.256586    8.381612   12.674552   10.517059    9.397854
2.094168
2      5.401607    9.911597    9.045255    9.788359   10.090470
1.683023
3      8.942805    9.601208   11.392682    9.694814    9.684365
3.292001
4      7.181162    9.846910   11.922439    9.217749    9.461191
5.110372
..          ...         ...         ...         ...         ...
...
796    4.484415    9.614701   12.031267    9.813063   10.092770
8.819269
797    6.555327    9.064002   11.633422   10.317266    8.745983
9.659081
798    3.589763    9.350636   12.180944   10.681194    9.466711
4.677458
799    4.745888    9.626383   11.198279   10.335513   10.400581
5.718751
800    9.139459   10.102934   11.641081   10.607358    9.844794
4.550716

     gene_20530
0      0.000000
1      0.000000
2      0.000000
3      0.000000
4      0.000000
..          ...
796    0.000000
797    0.000000
798    0.586693
799    0.000000
800    0.000000

[801 rows x 20531 columns]
```

```
#import T-SNE from sklearn
from sklearn.manifold import TSNE
m = TSNE(learning_rate=50)

tnse_features = m.fit_transform(df_tsne_data)
tnse_features[1:4,:]
```

```
array([[ -0.38058293,   5.7427387 ],
       [-41.13399   ,  -1.4808235 ],
       [-40.317463  ,  -1.0385877 ]], dtype=float32)
```

```
df_tsne_data['x'] = tnse_features[:,0]
df_tsne_data['y'] = tnse_features[:,1]
```

```
import seaborn as sns
sns.scatterplot(x='x',y='y',data=df_tsne_data)
plt.show()
```


```
df_tsne_data['cancer_type']=df_cat_data['Class']
sns.scatterplot(x='x',y='y',hue = 'cancer_type', data=df_tsne_data)
plt.show()
```

Dimensionality reduction using LDA

Reference - https://www.youtube.com/watch?v=azXCzI57Yfc

so doing LDA, because, there is a cluster which is distance and spread inside cluster. So LDA will explain both.

```
df_lda = master_data.drop(['Unnamed: 0'], axis=1)
df_lda = df_lda.drop(['Class'], axis=1)
x_lda = df_lda
x_lda
```

```
     gene_0    gene_1    gene_2    gene_3     gene_4  gene_5    gene_6
\
0       0.0  2.017209  3.265527  5.478487  10.431999     0.0  7.175175

1       0.0  0.592732  1.588421  7.586157   9.623011     0.0  6.816049

2       0.0  3.511759  4.327199  6.881787   9.870730     0.0  6.972130

3       0.0  3.663618  4.507649  6.659068  10.196184     0.0  7.843375

4       0.0  2.655741  2.821547  6.539454   9.738265     0.0  6.566967

..      ...       ...       ...       ...        ...     ...       ...

796     0.0  1.865642  2.718197  7.350099  10.006003     0.0  6.764792

797     0.0  3.942955  4.453807  6.346597  10.056868     0.0  7.320331

798     0.0  3.249582  3.707492  8.185901   9.504082     0.0  7.536589

799     0.0  2.590339  2.787976  7.318624   9.987136     0.0  9.213464

800     0.0  2.325242  3.805932  6.530246   9.560367     0.0  7.957027


       gene_7  gene_8  gene_9  ...  gene_20521  gene_20522  gene_20523
\
0    0.591871     0.0     0.0  ...    4.926711    8.210257    9.723516

1    0.000000     0.0     0.0  ...    4.593372    7.323865    9.740931

2    0.452595     0.0     0.0  ...    5.125213    8.127123   10.908640

3    0.434882     0.0     0.0  ...    6.076566    8.792959   10.141520

4    0.360982     0.0     0.0  ...    5.996032    8.891425   10.373790

..        ...     ...     ...  ...         ...         ...         ...
```

```
796  0.496922     0.0     0.0  ...    6.088133    9.118313   10.004852

797  0.000000     0.0     0.0  ...    6.371876    9.623335    9.823921

798  1.811101     0.0     0.0  ...    5.719386    8.610704   10.485517

799  0.000000     0.0     0.0  ...    5.785237    8.605387   11.004677

800  0.000000     0.0     0.0  ...    6.403075    8.594354   10.243079


     gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
0      7.220030    9.119813   12.003135    9.650743    8.921326
5.286759
1      6.256586    8.381612   12.674552   10.517059    9.397854
2.094168
2      5.401607    9.911597    9.045255    9.788359   10.090470
1.683023
3      8.942805    9.601208   11.392682    9.694814    9.684365
3.292001
4      7.181162    9.846910   11.922439    9.217749    9.461191
5.110372
..          ...         ...         ...         ...         ...
...
796    4.484415    9.614701   12.031267    9.813063   10.092770
8.819269
797    6.555327    9.064002   11.633422   10.317266    8.745983
9.659081
798    3.589763    9.350636   12.180944   10.681194    9.466711
4.677458
799    4.745888    9.626383   11.198279   10.335513   10.400581
5.718751
800    9.139459   10.102934   11.641081   10.607358    9.844794
4.550716

     gene_20530
0      0.000000
1      0.000000
2      0.000000
3      0.000000
4      0.000000
..          ...
796    0.000000
797    0.000000
798    0.586693
799    0.000000
800    0.000000
```

```
[801 rows x 20531 columns]
```

```
x_lda.shape
```

```
(801, 20531)
```

```
y_lda = master_data['Class']
y_lda.values
```

```
array([1, 2, 1, 1, 3, 1, 4, 1, 3, 1, 3, 4, 1, 3, 3, 3, 2, 4, 4, 1, 3,
4,
       2, 3, 4, 2, 5, 3, 3, 3, 3, 3, 4, 3, 1, 3, 4, 2, 3, 3, 4, 1, 1,
4,
       4, 3, 1, 5, 3, 2, 3, 2, 3, 1, 5, 3, 3, 5, 4, 3, 2, 4, 3, 2, 1,
5,
       3, 1, 4, 3, 4, 3, 3, 2, 3, 2, 3, 4, 1, 5, 3, 1, 3, 3, 1, 1, 3,
3,
       4, 3, 1, 1, 3, 3, 3, 1, 5, 3, 1, 3, 3, 4, 3, 4, 2, 4, 2, 5, 2,
2,
       1, 3, 2, 1, 3, 4, 4, 4, 3, 3, 2, 4, 2, 3, 1, 1, 1, 3, 4, 2, 5,
3,
       5, 3, 3, 4, 2, 3, 4, 5, 3, 1, 3, 4, 2, 5, 1, 3, 2, 2, 2, 2, 3,
3,
       2, 3, 3, 1, 1, 2, 1, 2, 4, 3, 1, 2, 5, 4, 2, 3, 4, 2, 3, 2, 3,
3,
       3, 1, 3, 4, 5, 4, 3, 1, 1, 1, 2, 2, 3, 2, 2, 4, 2, 1, 2, 3, 3,
3,
       2, 2, 3, 4, 4, 4, 4, 1, 3, 1, 3, 2, 2, 3, 1, 3, 1, 3, 3, 3, 2,
3,
       4, 2, 4, 4, 2, 3, 4, 1, 3, 2, 2, 1, 5, 3, 4, 1, 4, 5, 3, 4, 4,
2,
       1, 1, 2, 2, 4, 3, 3, 5, 3, 1, 5, 3, 1, 4, 1, 1, 1, 3, 5, 5, 2,
5,
       5, 1, 2, 3, 3, 4, 4, 3, 5, 1, 4, 1, 3, 3, 4, 3, 3, 3, 3, 2, 2,
3,
       3, 3, 4, 4, 4, 4, 3, 3, 3, 4, 3, 3, 2, 1, 3, 3, 5, 2, 1, 3, 3,
3,
       5, 3, 1, 3, 5, 2, 2, 1, 4, 3, 4, 4, 2, 5, 4, 3, 3, 3, 3, 4, 3,
3,
       1, 3, 4, 3, 2, 1, 4, 3, 1, 5, 3, 3, 3, 2, 2, 2, 3, 3, 1, 2, 3,
4,
       3, 5, 5, 2, 3, 4, 3, 3, 3, 5, 2, 5, 4, 1, 4, 3, 3, 4, 3, 5, 1,
2,
       1, 3, 4, 1, 3, 5, 4, 4, 5, 5, 1, 3, 3, 5, 4, 3, 1, 3, 3, 3, 2,
2,
       4, 2, 3, 4, 5, 1, 3, 2, 1, 3, 3, 3, 2, 3, 3, 1, 3, 1, 5, 3, 2,
3,
       3, 2, 3, 3, 3, 4, 2, 1, 3, 1, 4, 3, 4, 5, 3, 1, 2, 4, 3, 3, 4,
3,
       2, 3, 3, 1, 5, 3, 4, 2, 1, 3, 1, 3, 3, 3, 3, 4, 2, 3, 4, 3, 3,
```

```
2,
       2, 4, 5, 1, 5, 3, 4, 4, 3, 1, 4, 5, 2, 2, 3, 1, 1, 3, 1, 2, 4,
1,
       3, 2, 1, 2, 3, 3, 5, 2, 4, 5, 2, 3, 1, 3, 3, 1, 3, 5, 3, 5, 4,
3,
       3, 2, 2, 2, 5, 4, 2, 2, 3, 3, 4, 1, 2, 1, 3, 4, 3, 4, 4, 1, 1,
2,
       3, 4, 5, 5, 3, 4, 4, 3, 3, 1, 4, 5, 3, 3, 5, 2, 3, 3, 3, 4, 1,
2,
       2, 3, 4, 5, 4, 4, 3, 1, 2, 4, 3, 5, 2, 2, 2, 1, 2, 4, 3, 3, 5,
1,
       3, 3, 3, 4, 2, 2, 3, 1, 2, 2, 3, 4, 1, 5, 2, 1, 5, 2, 5, 4, 4,
3,
       3, 4, 4, 5, 3, 2, 1, 1, 4, 3, 2, 3, 3, 5, 3, 1, 1, 3, 3, 5, 3,
4,
       3, 3, 5, 3, 1, 3, 3, 4, 1, 2, 3, 3, 4, 3, 3, 3, 3, 3, 5, 2, 2,
3,
       3, 3, 1, 3, 3, 4, 2, 2, 4, 4, 2, 4, 5, 3, 5, 4, 3, 3, 1, 1, 1,
2,
       1, 1, 5, 3, 3, 5, 2, 4, 3, 4, 5, 3, 3, 3, 1, 2, 4, 3, 1, 4, 1,
3,
       2, 4, 1, 2, 1, 1, 3, 4, 1, 2, 5, 5, 3, 3, 3, 2, 4, 4, 4, 3, 2,
4,
       1, 3, 2, 1, 3, 1, 3, 4, 3, 5, 1, 1, 4, 1, 4, 3, 2, 2, 3, 3, 4,
3,
       3, 3, 3, 4, 4, 1, 5, 4, 3, 3, 4, 3, 2, 3, 3, 1, 3, 1, 3, 5, 3,
3,
       4, 2, 3, 3, 1, 3, 1, 5, 3, 3, 3, 2, 3, 2, 3, 2, 4, 4, 5, 5, 3,
3,
       3, 2, 3, 2, 4, 3, 2, 4, 2, 2, 2, 4, 3, 1, 3, 3, 4, 3, 1, 4, 3,
3,
       3, 3, 1, 2, 3, 2, 5, 1, 1])
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
as LDA
lda = LDA(n_components=2)
x_r2 = lda.fit(x_lda,y_lda).transform(x_lda)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/
discriminant_analysis.py:388: UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
```

```
lda.explained_variance_ratio_
```

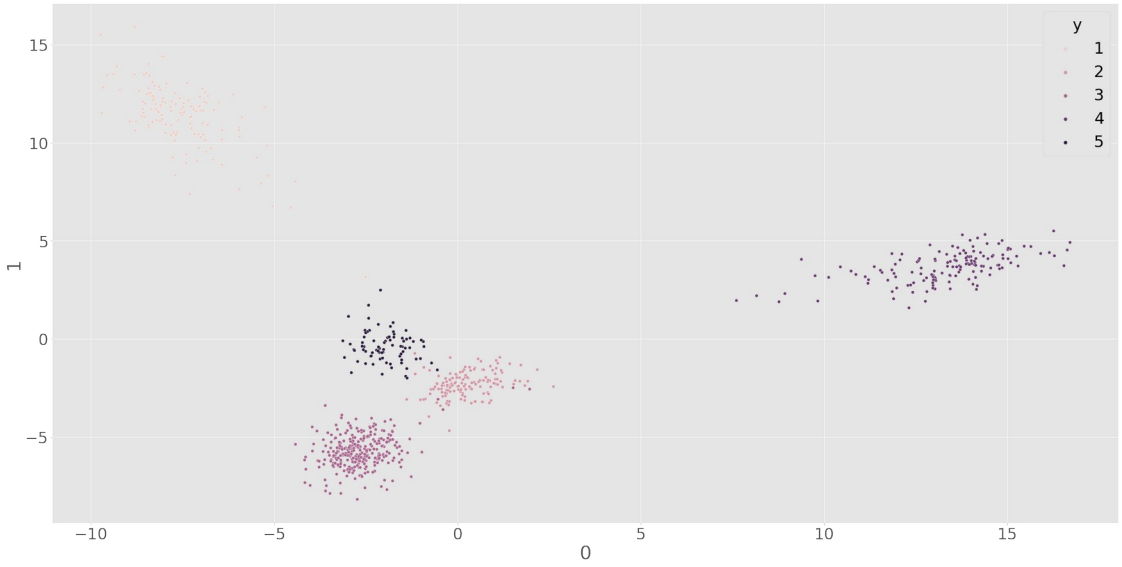
```
array([0.36219022, 0.30156109])
```

```
x_r3 = pd.DataFrame(data=x_r2)
x_r3['y']=y_lda
x_r3
```

```
            0          1  y
0   -7.958125  10.922818  1
1   -0.301563  -2.780898  2
2   -6.424952   8.870978  1
3   -6.934259  10.417199  1
4   -2.872004  -4.912284  3
..        ...        ... ..
796 -2.491183  -6.516482  3
797  0.217789  -1.859410  2
798 -1.426674  -0.474514  5
799 -7.800641  12.104337  1
800 -7.306312   7.388476  1

[801 rows x 3 columns]
```

```
sns.scatterplot(x=0,y=1,hue = 'y', data=x_r3)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f770b146490>
```



KMEANS Clustering with PCA = 2

```
from sklearn.cluster import KMeans
clusters = KMeans(5, n_init = 5)
clusters.fit(X_pca_with_2)
```

```
clusters.labels_
```

```
array([4, 1, 0, 0, 3, 1, 2, 1, 3, 0, 1, 2, 1, 4, 3, 3, 1, 2, 2, 0, 3,
1,
       1, 0, 2, 1, 0, 3, 1, 3, 3, 3, 2, 4, 0, 3, 2, 1, 1, 4, 2, 0, 1,
2,
       2, 3, 4, 0, 3, 1, 3, 0, 3, 1, 0, 0, 3, 0, 2, 1, 1, 1, 3, 1, 0,
0,
       1, 0, 2, 0, 2, 1, 0, 1, 3, 0, 3, 4, 1, 0, 3, 1, 3, 3, 4, 1, 3,
```

0,
2, 0, 0, 0, 3, 4, 3, 0, 0, 3, 3, 1, 3, 2, 3, 2, 1, 2, 0, 0, 0,
3,
4, 3, 1, 1, 3, 2, 2, 2, 3, 3, 1, 2, 1, 0, 1, 1, 0, 3, 2, 0, 0,
0,
4, 1, 3, 2, 2, 3, 2, 0, 3, 0, 3, 2, 1, 0, 4, 3, 1, 0, 1, 1, 3,
3,
1, 1, 3, 1, 0, 1, 4, 1, 2, 1, 1, 1, 0, 2, 1, 3, 2, 1, 3, 0, 3,
0,
1, 1, 3, 2, 0, 2, 3, 1, 0, 1, 1, 1, 3, 1, 1, 2, 4, 3, 1, 3, 4,
3,
0, 1, 3, 2, 2, 2, 2, 3, 4, 0, 1, 1, 1, 3, 1, 0, 1, 3, 3, 3, 1,
1,
2, 1, 2, 2, 1, 3, 2, 0, 3, 1, 4, 3, 0, 4, 2, 1, 2, 0, 1, 2, 2,
3,
0, 1, 1, 1, 2, 3, 3, 0, 3, 0, 1, 3, 1, 2, 4, 1, 1, 3, 1, 0, 3,
0,
0, 0, 3, 0, 3, 2, 2, 3, 0, 4, 2, 1, 3, 3, 2, 3, 3, 3, 4, 1, 1,
3,
3, 3, 2, 2, 2, 2, 4, 3, 3, 2, 3, 3, 1, 1, 1, 3, 0, 1, 0, 3, 3,
3,
4, 1, 3, 3, 0, 0, 1, 0, 2, 0, 2, 2, 3, 0, 1, 3, 3, 3, 3, 2, 3,
3,
0, 0, 2, 3, 1, 1, 2, 3, 0, 0, 1, 3, 3, 1, 4, 1, 3, 3, 1, 1, 3,
2,
3, 0, 1, 1, 3, 2, 3, 3, 0, 0, 1, 0, 1, 0, 2, 1, 3, 2, 3, 0, 0,
1,
0, 4, 2, 0, 3, 0, 2, 2, 0, 0, 0, 3, 4, 0, 2, 4, 1, 0, 3, 3, 1,
1,
2, 1, 3, 2, 0, 3, 3, 1, 1, 0, 4, 3, 1, 3, 0, 1, 1, 1, 0, 3, 3,
3,
3, 1, 3, 1, 3, 2, 1, 3, 3, 1, 2, 0, 2, 4, 3, 0, 1, 2, 3, 3, 2,
3,
1, 0, 3, 1, 4, 0, 2, 1, 4, 0, 0, 3, 3, 3, 3, 2, 3, 1, 2, 3, 3,
0,
1, 2, 0, 1, 0, 1, 2, 2, 0, 1, 2, 1, 1, 1, 3, 1, 1, 1, 3, 3, 2,
1,
1, 4, 3, 4, 3, 3, 0, 1, 2, 0, 1, 0, 0, 4, 1, 1, 3, 0, 3, 4, 2,
3,
3, 0, 3, 3, 0, 2, 1, 1, 3, 3, 1, 3, 0, 1, 3, 2, 0, 2, 2, 4, 1,
1,
1, 2, 0, 0, 3, 2, 2, 3, 1, 1, 2, 0, 1, 3, 0, 1, 3, 1, 3, 2, 3,
0,
1, 0, 2, 0, 2, 2, 0, 1, 0, 2, 3, 0, 0, 1, 1, 1, 1, 2, 3, 3, 0,
1,
3, 3, 3, 2, 1, 0, 0, 0, 0, 3, 3, 2, 0, 4, 1, 1, 0, 1, 4, 2, 2,
3,
3, 2, 2, 0, 3, 1, 0, 0, 2, 3, 1, 3, 1, 0, 1, 4, 1, 1, 3, 4, 3,
2,
3, 1, 1, 3, 1, 0, 3, 2, 0, 1, 3, 3, 2, 1, 3, 3, 3, 4, 0, 1, 1,

```
0,
       3, 3, 1, 3, 3, 2, 0, 3, 2, 2, 1, 2, 0, 1, 4, 2, 3, 3, 1, 1, 0,
1,
       0, 3, 0, 3, 3, 0, 0, 2, 3, 2, 0, 3, 3, 1, 3, 1, 2, 3, 3, 2, 0,
3,
       1, 2, 0, 1, 1, 1, 4, 2, 0, 1, 0, 0, 3, 1, 3, 1, 2, 2, 2, 1, 1,
2,
       1, 3, 1, 4, 3, 0, 3, 2, 1, 1, 1, 1, 2, 1, 2, 3, 1, 1, 3, 4, 2,
3,
       0, 3, 0, 2, 2, 1, 0, 2, 0, 4, 2, 3, 1, 3, 0, 3, 4, 0, 1, 0, 1,
3,
       2, 1, 3, 3, 1, 3, 0, 1, 3, 1, 3, 1, 3, 1, 1, 1, 2, 2, 0, 0, 3,
3,
       3, 0, 3, 1, 1, 1, 1, 2, 0, 1, 0, 2, 3, 4, 0, 0, 2, 3, 3, 1, 3,
3,
       3, 3, 3, 3, 3, 0, 1, 0, 1], dtype=int32)
```

```
pca_with_2_data_frame =
pd.DataFrame(data=X_pca_with_2,columns=['pca1','pca2'])
pca_with_2_data_frame.head()
```

```
        pca1       pca2
0 -57.446987  95.410981
1 -16.919430   0.732470
2 -70.345218 -19.303327
3 -49.161591  -9.227586
4 -18.132534 -51.327797
```

```
pca_with_2_data_frame['Cls_label'] = clusters.labels_
pca_with_2_data_frame['given_cancer_type'] = label.Class.values
pca_with_2_data_frame
```

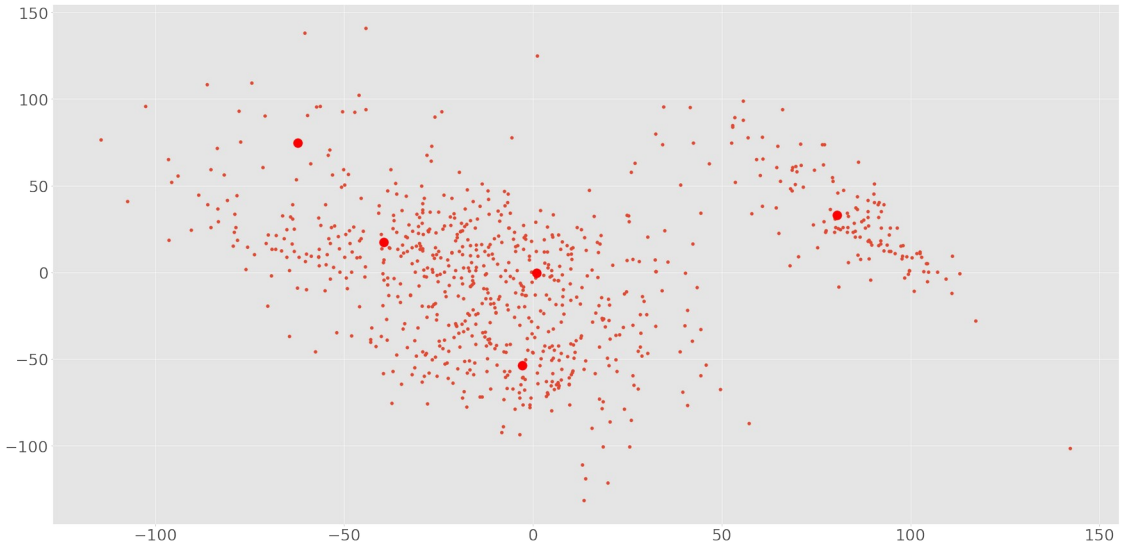
```
          pca1       pca2  Cls_label given_cancer_type
0   -57.446987  95.410981          4              PRAD
1   -16.919430   0.732470          1              LUAD
2   -70.345218 -19.303327          0              PRAD
3   -49.161591  -9.227586          0              PRAD
4   -18.132534 -51.327797          3              BRCA
..         ...        ...        ...               ...
796 -12.417385 -42.321574          3              BRCA
797 -29.415554  28.526281          0              LUAD
798  -4.133090  15.690014          1              COAD
799 -30.814757  33.526423          0              PRAD
800 -22.344557   4.052356          1              PRAD

[801 rows x 4 columns]
```

```
clusters.cluster_centers_
```

```
array([[-45.20369136,  15.07808379],
       [ -2.06263931,   6.32719153],
```

```
       [ 80.93171918,  32.80898515],
       [ -2.18176324, -51.4499728 ],
       [-60.32075428,  74.65389731]])
```

```
kmeans = KMeans(n_clusters=5, init='k-means++', max_iter=300,
n_init=10, random_state=0)
pred_y = kmeans.fit_predict(X_pca_with_2)
plt.scatter(X_pca_with_2[:,0], X_pca_with_2[:,1])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s=300, c='red')
plt.show()
```



KMEANS Clustering with PCA = .995

```
from sklearn.cluster import KMeans
clusters_995 = KMeans(5, n_init = 5)
clusters_995.fit(X_pca_with_995)
clusters_995.labels_
```

```
array([4, 0, 4, 4, 3, 4, 1, 4, 3, 4, 3, 1, 4, 3, 3, 3, 0, 1, 1, 4, 3,
1,
       0, 3, 1, 0, 2, 3, 3, 3, 3, 3, 1, 3, 4, 3, 1, 0, 3, 3, 1, 4, 4,
1,
       1, 3, 4, 2, 3, 0, 3, 0, 3, 4, 2, 3, 3, 2, 1, 3, 0, 3, 3, 0, 4,
2,
       3, 4, 1, 3, 1, 3, 3, 0, 3, 0, 3, 1, 4, 2, 3, 4, 3, 3, 4, 4, 3,
3,
       1, 3, 4, 4, 3, 3, 3, 4, 2, 3, 4, 3, 3, 1, 3, 1, 0, 1, 0, 2, 0,
0,
       4, 3, 0, 4, 3, 1, 1, 1, 3, 3, 0, 1, 0, 3, 4, 4, 4, 3, 1, 3, 2,
3,
       2, 3, 3, 1, 0, 3, 1, 2, 3, 4, 3, 1, 0, 2, 4, 3, 0, 0, 0, 0, 3,
3,
```

0, 3, 3, 4, 4, 0, 4, 0, 1, 3, 4, 0, 2, 1, 0, 3, 1, 0, 3, 0, 3,
3,
3, 4, 3, 1, 2, 1, 3, 4, 4, 4, 0, 0, 3, 0, 0, 1, 0, 4, 0, 3, 3,
3,
0, 0, 3, 1, 1, 1, 1, 4, 3, 4, 3, 0, 0, 3, 4, 3, 4, 3, 3, 3, 0,
3,
1, 0, 1, 1, 0, 3, 1, 4, 3, 0, 0, 4, 2, 3, 1, 4, 1, 2, 3, 1, 1,
0,
4, 4, 0, 0, 1, 3, 3, 2, 3, 4, 0, 3, 4, 1, 4, 4, 4, 3, 2, 2, 0,
2,
2, 4, 0, 3, 3, 1, 1, 3, 2, 4, 1, 4, 3, 3, 1, 3, 3, 3, 3, 0, 0,
3,
3, 3, 1, 1, 1, 1, 3, 3, 3, 1, 3, 3, 0, 4, 3, 3, 2, 0, 4, 3, 3,
3,
2, 3, 4, 3, 2, 0, 0, 4, 1, 3, 1, 1, 0, 2, 1, 3, 3, 3, 3, 1, 3,
3,
4, 3, 1, 3, 0, 4, 1, 3, 4, 2, 3, 3, 3, 0, 0, 0, 3, 3, 4, 0, 3,
1,
3, 2, 2, 0, 3, 1, 3, 3, 3, 2, 0, 2, 1, 4, 1, 3, 3, 1, 3, 2, 4,
0,
4, 3, 1, 4, 3, 2, 1, 1, 2, 2, 4, 3, 3, 2, 1, 0, 4, 3, 3, 3, 0,
0,
1, 0, 3, 1, 2, 4, 3, 0, 4, 3, 3, 3, 0, 3, 3, 4, 3, 4, 2, 3, 0,
3,
3, 0, 3, 3, 3, 1, 0, 4, 3, 4, 1, 3, 1, 2, 3, 4, 0, 1, 3, 3, 1,
3,
0, 3, 3, 4, 2, 3, 1, 0, 4, 3, 4, 3, 3, 3, 3, 1, 0, 3, 1, 3, 3,
0,
0, 1, 2, 4, 2, 3, 1, 1, 3, 4, 1, 2, 0, 0, 3, 4, 4, 3, 4, 0, 1,
4,
3, 0, 4, 0, 3, 3, 2, 0, 1, 2, 0, 3, 4, 3, 3, 4, 3, 2, 3, 2, 1,
3,
3, 0, 0, 0, 2, 1, 0, 0, 3, 3, 1, 4, 0, 4, 3, 1, 3, 1, 1, 4, 4,
0,
3, 1, 2, 2, 3, 1, 1, 3, 3, 4, 1, 2, 3, 3, 2, 0, 3, 3, 3, 1, 4,
0,
0, 3, 1, 2, 1, 1, 3, 4, 0, 1, 3, 2, 0, 0, 0, 4, 0, 1, 3, 3, 2,
4,
3, 3, 3, 1, 0, 0, 3, 4, 0, 0, 3, 1, 4, 2, 0, 4, 2, 0, 2, 1, 1,
3,
3, 1, 1, 2, 3, 3, 4, 4, 1, 3, 0, 3, 3, 2, 3, 4, 4, 3, 3, 2, 3,
1,
3, 3, 2, 3, 4, 3, 3, 1, 4, 0, 3, 3, 1, 3, 3, 3, 3, 3, 2, 0, 0,
3,
3, 3, 4, 3, 3, 1, 0, 0, 1, 1, 0, 1, 2, 3, 2, 1, 3, 3, 4, 4, 4,
0,
4, 4, 2, 3, 3, 2, 0, 1, 3, 1, 2, 3, 3, 3, 4, 0, 1, 3, 4, 1, 4,
3,
0, 1, 4, 0, 4, 4, 3, 1, 4, 0, 2, 2, 3, 3, 3, 0, 1, 1, 1, 3, 0,
1,

```
       4, 3, 0, 4, 3, 4, 3, 1, 3, 2, 4, 4, 1, 4, 1, 3, 0, 0, 3, 3, 1,
3,
       3, 3, 3, 1, 1, 4, 2, 1, 3, 3, 1, 3, 0, 3, 3, 4, 3, 4, 3, 2, 3,
3,
       1, 0, 3, 3, 4, 3, 4, 2, 3, 3, 3, 0, 3, 0, 3, 0, 1, 1, 2, 2, 3,
3,
       3, 0, 3, 0, 1, 3, 0, 1, 0, 0, 0, 1, 3, 4, 3, 3, 1, 3, 4, 1, 3,
3,
       3, 3, 4, 0, 3, 0, 0, 4, 4], dtype=int32)
```

```
pca_with_995_data_frame = pd.DataFrame(data=X_pca_with_995)
pca_with_995_data_frame.head()
pca_with_995_data_frame['Cls_label'] = clusters.labels_
pca_with_995_data_frame['given_cancer_type'] = label.Class.values

pca_with_995_data_frame.head(5)
```

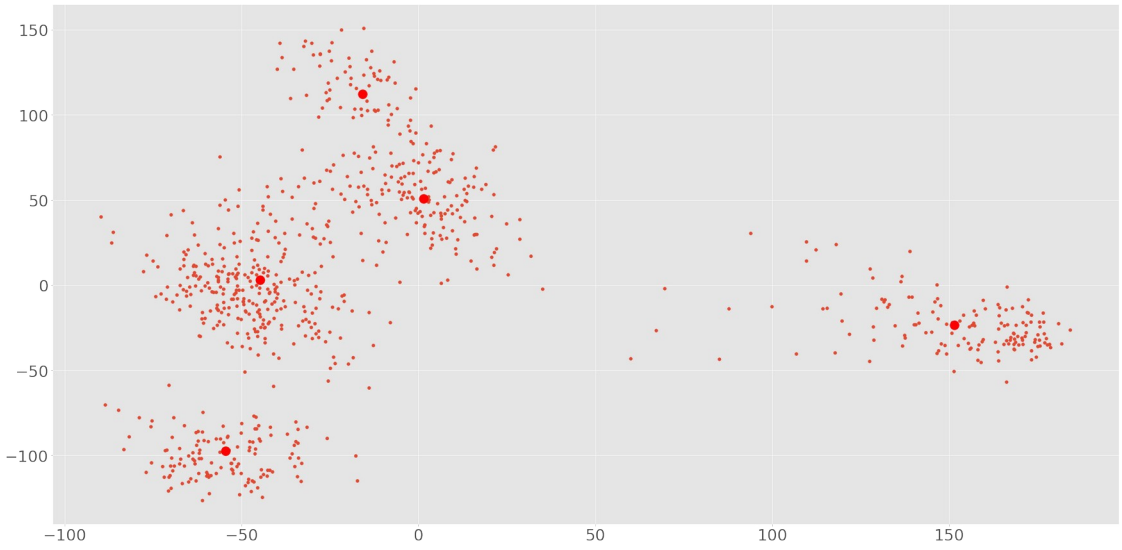
```
           0          1          2          3          4
5  \
0 -62.755415 -94.071973  89.519831 -15.942567  81.423539  -13.998292

1  -2.432896  90.585842  -1.067308 -53.083120 -15.676684   60.842472

2 -71.266853  -8.064608  66.112455  81.381475  -7.525685  109.824273

3 -84.770785 -73.244566  74.181000  27.022697 -18.044895   50.116433

4 -69.560171  -9.612940 -67.497549  34.868543  -1.795849   -6.676780


           6          7          8          9  ...       739       740
\
0   7.716073 -22.936551 -32.837892  -2.202680  ... -0.626193 -1.265756

1  10.257369 -48.822959  14.257400 -12.214352  ... -0.593678 -0.403462

2   5.519407 -13.364480  38.415728  -5.124731  ...  0.328453  0.004078

3  -3.495197 -11.318520   8.319656  -3.149509  ...  0.652455 -3.624900

4  -2.840781  16.780157 -49.319753  10.508631  ...  2.767486 -0.631562


        741       742       743       744       745       746
Cls_label  \
0 -0.017984 -2.740860  0.944037  3.092581  0.713598 -0.082122
4
1  1.181537  0.490910  0.197768  0.013967 -0.395176 -0.949947
1
2  0.363928 -1.109210  0.331488  0.128899 -0.264530  0.384594
```

```
0
3 -1.203028 -2.347912  1.577992 -0.781748  0.120442 -0.057973
0
4 -0.794275 -0.514008 -1.875969 -2.526109 -1.073803 -1.161728
3

   given_cancer_type
0               PRAD
1               LUAD
2               PRAD
3               PRAD
4               BRCA

[5 rows x 749 columns]
```

```
pca_with_995_data_frame.shape
```

```
(801, 749)
```

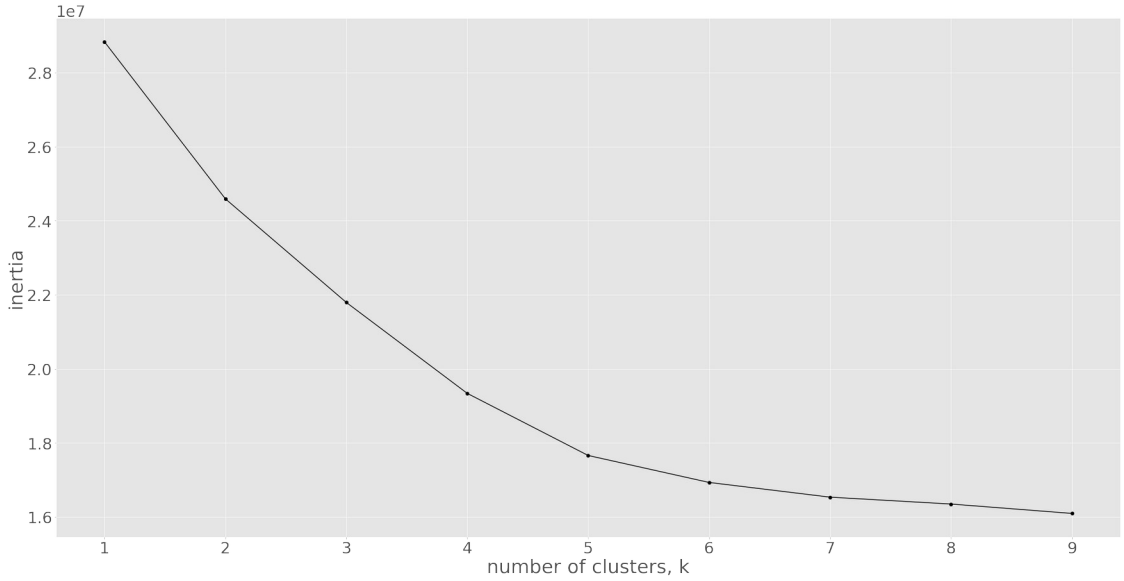
```
kmeans = KMeans(n_clusters=5, init='k-means++', max_iter=300,
n_init=10, random_state=0)
pred_y = kmeans.fit_predict(X_pca_with_995)
plt.scatter(X_pca_with_995[:,0], X_pca_with_995[:,1])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s=300, c='red')
plt.show()
```



Elbow method

```
ks = range(1, 10)
inertias = []
for k in ks:
    # Create a KMeans instance with k clusters: model
```

```
    model = KMeans(n_clusters=k)

    # Fit model to samples
    model.fit(df_pca_995)

    # Append the inertia to the list of inertias
    inertias.append(model.inertia_)

plt.plot(ks, inertias, '-o', color='black')
plt.xlabel('number of clusters, k')
plt.ylabel('inertia')
plt.xticks(ks)
plt.show()
```



Build Classification Models

BUILDING AND RUNNING MANY ALGORITHMS AT ONCE

Reference - https://www.youtube.com/watch?v=7uLzGRlXXDw&list=PL9z-nia3KLoTBbK5ATeZnWMejwnIlVEPa&index=11

```
ml_x = x_lda
ml_y = y_lda
ml_x.shape,ml_y.shape
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =
train_test_split(ml_x,ml_y,test_size=0.30,random_state=30)

ml_x.head(5)
```

```
   gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
gene_6  \
0     0.0  2.017209  3.265527  5.478487  10.431999     0.0  7.175175
```

```
1     0.0  0.592732  1.588421  7.586157   9.623011     0.0  6.816049

2     0.0  3.511759  4.327199  6.881787   9.870730     0.0  6.972130

3     0.0  3.663618  4.507649  6.659068  10.196184     0.0  7.843375

4     0.0  2.655741  2.821547  6.539454   9.738265     0.0  6.566967


     gene_7  gene_8  gene_9  ...  gene_20521  gene_20522
gene_20523  \
0  0.591871     0.0     0.0  ...    4.926711    8.210257    9.723516

1  0.000000     0.0     0.0  ...    4.593372    7.323865    9.740931

2  0.452595     0.0     0.0  ...    5.125213    8.127123   10.908640

3  0.434882     0.0     0.0  ...    6.076566    8.792959   10.141520

4  0.360982     0.0     0.0  ...    5.996032    8.891425   10.373790


   gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
0    7.220030    9.119813   12.003135    9.650743    8.921326
5.286759
1    6.256586    8.381612   12.674552   10.517059    9.397854
2.094168
2    5.401607    9.911597    9.045255    9.788359   10.090470
1.683023
3    8.942805    9.601208   11.392682    9.694814    9.684365
3.292001
4    7.181162    9.846910   11.922439    9.217749    9.461191
5.110372

   gene_20530
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0

[5 rows x 20531 columns]

ml_y.head(5)

0    1
1    2
2    1
```

```
3    1
4    3
Name: Class, dtype: int64
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import f1_score
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import make_scorer
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score, classification_report, \
            precision_score, recall_score, f1_score, roc_auc_score,
roc_curve # ConfusionMatrixDisplay
```

```
models = {
    "Decision Tree": DecisionTreeClassifier(max_depth=5),
    "Support Vector Classifier":SVC(probability=True, kernel='rbf'),
    "Random Forest Classifier":
RandomForestClassifier(n_estimators=100),
    "Gaussian Naive Bayes": GaussianNB(),
    "K-Nearest Neighbour": KNeighborsClassifier(n_neighbors=5)
}
```

```
for i in range(len(list(models))):
  print(i)
  model = list(models.values())[i]
  model.fit(ml_x, ml_y) # train on LDA dataframe

  # Make predictions
  y_train_pred = model.predict(x_train)
  y_test_pred = model.predict(x_test)

  # Traning set performance
  model_train_accuracy = accuracy_score(y_train, y_train_pred)
  model_train_f1 = f1_score(y_train, y_train_pred, average='weighted')
  model_train_precision = precision_score(y_train, y_train_pred,
average='micro')
  model_train_recall = recall_score(y_train, y_train_pred,
average='micro')


 # Testing set performance
```

```
  model_test_accuracy = accuracy_score(y_test, y_test_pred)
  model_test_f1 = f1_score(y_test, y_test_pred, average='weighted')
  model_test_precision = precision_score(y_test, y_test_pred,
average='micro')
  model_test_recall = recall_score(y_test, y_test_pred,
average='micro')

  print(list(models.keys())[i])

  # Model Train
  print('Model performance for Traning set')
  print("- Accuracy: {:.4f}".format(model_train_accuracy))
  print("- F1 Score: {:.4f}".format(model_train_f1))
  print("- Precision: {:.4f}".format(model_train_precision))
  print("- Recall: {:.4f}".format(model_train_recall))


print("----------------------------------------------------------------
")

  # Model Test
  print('Model performance for Testing set')
  print("- Accuracy: {:.4f}".format(model_test_accuracy))
  print("- F1 Score: {:.4f}".format(model_test_f1))
  print("- Precision: {:.4f}".format(model_test_precision))
  print("- Recall: {:.4f}".format(model_test_recall))


  print("="*35)
  print("\n")
```

```
0
Decision Tree
Model performance for Traning set
- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000
----------------------------------------------------------------
Model performance for Testing set
- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000
===================================


1
Support Vector Classifier
Model performance for Traning set
```

- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000

Model performance for Testing set
- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000
==========

2
Random Forest Classifier
Model performance for Traning set
- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000

Model performance for Testing set
- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000
==========

3
Gaussian Naive Bayes
Model performance for Traning set
- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000

Model performance for Testing set
- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000
==========

4
K-Nearest Neighbour
Model performance for Traning set
- Accuracy: 1.0000
- F1 Score: 1.0000

```
- Precision: 1.0000
- Recall: 1.0000
----------------------------------------------------------------
Model performance for Testing set
- Accuracy: 0.9959
- F1 Score: 0.9958
- Precision: 0.9959
- Recall: 0.9959
==================================
```

```
rf_params = {
             "max_depth": [5, 8, 15, None, 10],
             "max_features": [5, 7, "auto", 8],
             "min_samples_split": [2, 8, 15, 20],
             "n_estimators": [100, 200, 500, 1000],
             }

rf_params
```

```
{'max_depth': [5, 8, 15, None, 10],
 'max_features': [5, 7, 'auto', 8],
 'min_samples_split': [2, 8, 15, 20],
 'n_estimators': [100, 200, 500, 1000]}
```

```
randomcv_models = [
                  ("RF", RandomForestClassifier(), rf_params)

                   ]

randomcv_models
```

```
[('RF',
  RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',
                         max_depth=None, max_features='auto',
max_leaf_nodes=None,
                         min_impurity_decrease=0.0,
min_impurity_split=None,
                         min_samples_leaf=1, min_samples_split=2,
                         min_weight_fraction_leaf=0.0,
n_estimators='warn',
                         n_jobs=None, oob_score=False,
random_state=None,
                         verbose=0, warm_start=False),
  {'max_depth': [5, 8, 15, None, 10],
   'max_features': [5, 7, 'auto', 8],
   'min_samples_split': [2, 8, 15, 20],
   'n_estimators': [100, 200, 500, 1000]})]
```

```
from sklearn.model_selection import RandomizedSearchCV

model_param = {}
for name, model, params in randomcv_models:
  random = RandomizedSearchCV(estimator=model,
                              param_distributions=params,
                              n_iter=100,
                              cv=3,
                              verbose=2,
                              n_jobs=-1)
  random.fit(x_train, y_train)
  model_param[name] = random.best_params_

for model_name in model_param:
  print(f"--------------- Best param for {model_name}
---------------")
  print(model_param[model_name])
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 158 tasks      | elapsed:  6.7min
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed: 10.5min finished

--------------- Best param for RF ---------------
{'n_estimators': 100, 'min_samples_split': 20, 'max_features': 'auto',
'max_depth': 15}
```

```
models = {
    "Random Forest": RandomForestClassifier(n_estimators=1000,
min_samples_split = 15 ,
                                            max_features = "auto",
max_depth = None)
}

for i in range(len(list(models))):
  model = list(models.values())[i]
  model.fit(x_train, y_train) # Train Model

  # Make predictions
  y_train_pred = model.predict(x_train)
  y_test_pred = model.predict(x_test)


  # Training set performance
  model_train_accuracy = accuracy_score(y_train, y_train_pred)
  model_train_f1 = f1_score(y_train, y_train_pred, average='weighted')
  model_train_precision = precision_score(y_train, y_train_pred,
```

```
average='micro')
  model_train_recall = recall_score(y_train, y_train_pred,
average='micro')

  # Testing set performance
  model_test_accuracy = accuracy_score(y_test, y_test_pred)
  model_test_f1 = f1_score(y_test, y_test_pred, average='weighted')
  model_test_precision = precision_score(y_test, y_test_pred,
average='micro')
  model_test_recall = recall_score(y_test, y_test_pred,
average='micro')

  print(list(models.keys())[i])

  # Model Train
  print('Model performance for Traning set')
  print("- Accuracy: {:.4f}".format(model_train_accuracy))
  print("- F1 Score: {:.4f}".format(model_train_f1))
  print("- Precision: {:.4f}".format(model_train_precision))
  print("- Recall: {:.4f}".format(model_train_recall))


print("----------------------------------------------------------------
")

  # Model Test
  print('Model performance for Testing set')
  print("- Accuracy: {:.4f}".format(model_test_accuracy))
  print("- F1 Score: {:.4f}".format(model_test_f1))
  print("- Precision: {:.4f}".format(model_test_precision))
  print("- Recall: {:.4f}".format(model_test_recall))


  print("="*35)
  print("\n")
```

```
Random Forest
Model performance for Traning set
- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000
----------------------------------------------------------------
Model performance for Testing set
- Accuracy: 0.9917
- F1 Score: 0.9917
- Precision: 0.9917
- Recall: 0.9917
===================================
```

DEEP NEURAL NETWORK

```
!wget
https://www.dropbox.com/sh/8q39v4rvo9hq7hy/AAAfAs9J12eevM_9_jPySJ1xa?
dl=0

--2022-09-17 10:03:50--
https://www.dropbox.com/sh/8q39v4rvo9hq7hy/AAAfAs9J12eevM_9_jPySJ1xa?
dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.85.18,
2620:100:6035:18::a27d:5512
Connecting to www.dropbox.com (www.dropbox.com)|162.125.85.18|:443...
connected.
HTTP request sent, awaiting response... 302 Found
Location: /sh/raw/8q39v4rvo9hq7hy/AAAfAs9J12eevM_9_jPySJ1xa
[following]
--2022-09-17 10:03:51--
https://www.dropbox.com/sh/raw/8q39v4rvo9hq7hy/AAAfAs9J12eevM_9_jPySJ1
xa
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location:
https://uca72f22e353d4299b595d7a3e63.dl.dropboxusercontent.com/zip_dow
nload_get/
BQi15JCy82HYMxgiUSypUHWMIaJLq7E6HIRod7ybXZfL_JzGxOXnHyWW9FxAJMuFjNAlmq
Ej_GARKWjUnwYCtnAYwN1EObKXO-5METrblkDmLw# [following]
--2022-09-17 10:03:51--
https://uca72f22e353d4299b595d7a3e63.dl.dropboxusercontent.com/zip_dow
nload_get/
BQi15JCy82HYMxgiUSypUHWMIaJLq7E6HIRod7ybXZfL_JzGxOXnHyWW9FxAJMuFjNAlmq
Ej_GARKWjUnwYCtnAYwN1EObKXO-5METrblkDmLw
Resolving uca72f22e353d4299b595d7a3e63.dl.dropboxusercontent.com
(uca72f22e353d4299b595d7a3e63.dl.dropboxusercontent.com)...
162.125.81.15, 2620:100:6035:15::a27d:550f
Connecting to uca72f22e353d4299b595d7a3e63.dl.dropboxusercontent.com
(uca72f22e353d4299b595d7a3e63.dl.dropboxusercontent.com)|
162.125.81.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 206199237 (197M) [application/zip]
Saving to: ‘AAAfAs9J12eevM_9_jPySJ1xa?dl=0.2’

AAAfAs9J12eevM_9_jP 100%[===================>] 196.65M  11.3MB/s    in
14s

2022-09-17 10:04:06 (14.2 MB/s) - ‘AAAfAs9J12eevM_9_jPySJ1xa?dl=0.2’
```

```
saved [206199237/206199237]
```

```
!unzip -qq /content/AAAfAs9J12eevM_9_jPySJ1xa?dl=0
```

```
warning:  stripped absolute path spec from /
mapname:  conversion of  failed
replace data.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
replace labels.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y

1 archive had fatal errors.
```

```
!pip install scikit-learn==0.21.2
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn==0.21.2 in
/usr/local/lib/python3.7/dist-packages (0.21.2)
Requirement already satisfied: joblib>=0.11 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn==0.21.2)
(1.1.0)
Requirement already satisfied: numpy>=1.11.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn==0.21.2)
(1.21.6)
Requirement already satisfied: scipy>=0.17.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn==0.21.2)
(1.4.1)
```

```
!pip install tensorflow==2.2.0
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow==2.2.0 in
/usr/local/lib/python3.7/dist-packages (2.2.0)
Requirement already satisfied: numpy<2.0,>=1.16.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.21.6)
Requirement already satisfied: scipy==1.4.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.4.1)
Requirement already satisfied: absl-py>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.2.0)
Requirement already satisfied: gast==0.3.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(0.3.3)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.15.0)
Requirement already satisfied: tensorboard<2.3.0,>=2.2.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
```

```
(2.2.2)
Requirement already satisfied: wheel>=0.26 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(0.37.1)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.1.0)
Requirement already satisfied: astunparse==1.6.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.6.3)
Requirement already satisfied: google-pasta>=0.1.8 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(0.2.0)
Requirement already satisfied: h5py<2.11.0,>=2.10.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(2.10.0)
Requirement already satisfied: protobuf>=3.8.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(3.17.3)
Requirement already satisfied: grpcio>=1.8.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.48.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(3.3.0)
Requirement already satisfied: wrapt>=1.11.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.14.1)
Requirement already satisfied: tensorflow-estimator<2.3.0,>=2.2.0
in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(2.2.0)
Requirement already satisfied: keras-preprocessing>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0)
(1.1.2)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.8.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (2.23.0)
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (57.4.0)
Requirement already satisfied: google-auth<2,>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.35.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.4.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
```

```
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (0.4.6)
Requirement already satisfied: werkzeug>=0.11.15 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.0.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.9)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.2.4)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
(1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in
/usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.12.0)
Requirement already satisfied: typing-extensions>=3.6.4 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
(4.1.1)
Requirement already satisfied: zipp>=0.5 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
(3.8.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<2,>=1.6.3->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
(0.4.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (2022.6.15)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.3.0,>=2.2.0-
>tensorflow==2.2.0) (3.2.0)
```

```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
colors = ['royalblue','red','deeppink', 'maroon', 'mediumorchid',
'tan', 'forestgreen', 'olive', 'goldenrod', 'lightcyan', 'navy']
vectorizer = np.vectorize(lambda x: colors[x % len(colors)])

import warnings
warnings.filterwarnings(action='ignore',category=DeprecationWarning)
warnings.filterwarnings(action='ignore',category=FutureWarning)

import sys
import csv

csv.field_size_limit(sys.maxsize)
```

```
9223372036854775807
```

```
import tensorflow as tf
tf.test.gpu_device_name()
```

```
{"type":"string"}
```

```
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

```
[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 5245237932281739182, name: "/device:XLA_CPU:0"
 device_type: "XLA_CPU"
 memory_limit: 17179869184
 locality {
 }
 incarnation: 17797237073633515140
 physical_device_desc: "device: XLA_CPU device"]
```

```
!cat /proc/meminfo
```

```
MemTotal:       13297220 kB
MemFree:         5183100 kB
MemAvailable:    7943636 kB
Buffers:           67868 kB
Cached:          2449300 kB
```

```
SwapCached:            0 kB
Active:          3357100 kB
Inactive:        4536088 kB
Active(anon):    2658640 kB
Inactive(anon):  2281216 kB
Active(file):     698460 kB
Inactive(file):  2254872 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:              2968 kB
Writeback:             0 kB
AnonPages:       5375548 kB
Mapped:           481072 kB
Shmem:             22208 kB
KReclaimable:     101660 kB
Slab:             136344 kB
SReclaimable:     101660 kB
SUnreclaim:        34684 kB
KernelStack:        4992 kB
PageTables:        25908 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     6648608 kB
Committed_AS:   10640636 kB
VmallocTotal:   34359738367 kB
VmallocUsed:        9812 kB
VmallocChunk:          0 kB
Percpu:             1408 kB
HardwareCorrupted:     0 kB
AnonHugePages:    731136 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:      164672 kB
DirectMap2M:    10317824 kB
DirectMap1G:     5242880 kB
```

```
label = pd.read_csv('/content/labels.csv', delimiter=',',
engine='python')
data = pd.read_csv('/content/data.csv', delimiter=',',
engine='python')
data.describe()
```

```
           gene_0      gene_1      gene_2      gene_3      gene_4
gene_5  \
count  801.000000  801.000000  801.000000  801.000000  801.000000
801.0
mean     0.026642    3.010909    3.095350    6.722305    9.813612
0.0
std      0.136850    1.200828    1.065601    0.638819    0.506537
0.0
min      0.000000    0.000000    0.000000    5.009284    8.435999
0.0
25%      0.000000    2.299039    2.390365    6.303346    9.464466
0.0
50%      0.000000    3.143687    3.127006    6.655893    9.791599
0.0
75%      0.000000    3.883484    3.802534    7.038447   10.142324
0.0
max      1.482332    6.237034    6.063484   10.129528   11.355621
0.0

           gene_6      gene_7      gene_8      gene_9  ...  gene_20521
\
count  801.000000  801.000000  801.000000  801.000000  ...  801.000000

mean     7.405509    0.499882    0.016744    0.013428  ...    5.896573

std      1.108237    0.508799    0.133635    0.204722  ...    0.746399

min      3.930747    0.000000    0.000000    0.000000  ...    2.853517

25%      6.676042    0.000000    0.000000    0.000000  ...    5.454926

50%      7.450114    0.443076    0.000000    0.000000  ...    5.972582

75%      8.121984    0.789354    0.000000    0.000000  ...    6.411292

max     10.718190    2.779008    1.785592    4.067604  ...    7.771054


       gene_20522  gene_20523  gene_20524  gene_20525  gene_20526
gene_20527  \
count  801.000000  801.000000  801.000000  801.000000  801.000000
801.000000
mean     8.765891   10.056252    4.847727    9.741987   11.742228
```

```
10.155271
std      0.603176    0.379278    2.382728    0.533898    0.670371
0.580569
min      6.678368    8.669456    0.000000    7.974942    9.045255
7.530141
25%      8.383834    9.826027    3.130750    9.400747   11.315857
9.836525
50%      8.784144   10.066385    5.444935    9.784524   11.749802
10.191207
75%      9.147136   10.299025    6.637412   10.082269   12.177852
10.578561
max     11.105431   11.318243    9.207495   11.811632   13.715361
11.675653

       gene_20528  gene_20529  gene_20530
count  801.000000  801.000000  801.000000
mean     9.590726    5.528177    0.095411
std      0.563849    2.073859    0.364529
min      7.864533    0.593975    0.000000
25%      9.244219    4.092385    0.000000
50%      9.566511    5.218618    0.000000
75%      9.917888    6.876382    0.000000
max     12.813320   11.205836    5.254133

[8 rows x 20531 columns]
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense
from tensorflow.keras.regularizers import l2
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Dropout
```

```
# Merge both the datasets
master_data = pd.merge(data, label)
master_data.head()
```

```
  Unnamed: 0  gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
\
0   sample_0     0.0  2.017209  3.265527  5.478487  10.431999     0.0

1   sample_1     0.0  0.592732  1.588421  7.586157   9.623011     0.0

2   sample_2     0.0  3.511759  4.327199  6.881787   9.870730     0.0

3   sample_3     0.0  3.663618  4.507649  6.659068  10.196184     0.0
```

```
4   sample_4     0.0  2.655741  2.821547  6.539454   9.738265     0.0


     gene_6    gene_7  gene_8  ...  gene_20522  gene_20523  gene_20524
\
0  7.175175  0.591871     0.0  ...    8.210257    9.723516    7.220030

1  6.816049  0.000000     0.0  ...    7.323865    9.740931    6.256586

2  6.972130  0.452595     0.0  ...    8.127123   10.908640    5.401607

3  7.843375  0.434882     0.0  ...    8.792959   10.141520    8.942805

4  6.566967  0.360982     0.0  ...    8.891425   10.373790    7.181162


   gene_20525  gene_20526  gene_20527  gene_20528  gene_20529
gene_20530  \
0    9.119813   12.003135    9.650743    8.921326    5.286759
0.0
1    8.381612   12.674552   10.517059    9.397854    2.094168
0.0
2    9.911597    9.045255    9.788359   10.090470    1.683023
0.0
3    9.601208   11.392682    9.694814    9.684365    3.292001
0.0
4    9.846910   11.922439    9.217749    9.461191    5.110372
0.0

   Class
0   PRAD
1   LUAD
2   PRAD
3   PRAD
4   BRCA

[5 rows x 20533 columns]

deep_learning_data = master_data

features=deep_learning_data.drop(['Unnamed: 0'],axis=1)
features=features.drop(['Class'],axis=1)
target=deep_learning_data['Class']
features.head(5)

   gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
gene_6  \
0     0.0  2.017209  3.265527  5.478487  10.431999     0.0  7.175175
```

```
1     0.0  0.592732  1.588421  7.586157   9.623011     0.0  6.816049

2     0.0  3.511759  4.327199  6.881787   9.870730     0.0  6.972130

3     0.0  3.663618  4.507649  6.659068  10.196184     0.0  7.843375

4     0.0  2.655741  2.821547  6.539454   9.738265     0.0  6.566967


     gene_7  gene_8  gene_9  ...  gene_20521  gene_20522
gene_20523  \
0  0.591871     0.0     0.0  ...    4.926711    8.210257    9.723516

1  0.000000     0.0     0.0  ...    4.593372    7.323865    9.740931

2  0.452595     0.0     0.0  ...    5.125213    8.127123   10.908640

3  0.434882     0.0     0.0  ...    6.076566    8.792959   10.141520

4  0.360982     0.0     0.0  ...    5.996032    8.891425   10.373790


   gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
0    7.220030    9.119813   12.003135    9.650743    8.921326
5.286759
1    6.256586    8.381612   12.674552   10.517059    9.397854
2.094168
2    5.401607    9.911597    9.045255    9.788359   10.090470
1.683023
3    8.942805    9.601208   11.392682    9.694814    9.684365
3.292001
4    7.181162    9.846910   11.922439    9.217749    9.461191
5.110372

   gene_20530
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0

[5 rows x 20531 columns]

target.head(5)

0    PRAD
1    LUAD
2    PRAD
```

```
3    PRAD
4    BRCA
Name: Class, dtype: object

x=features
x.head(5)
```

```
   gene_0    gene_1    gene_2    gene_3     gene_4  gene_5
gene_6  \
0     0.0  2.017209  3.265527  5.478487  10.431999     0.0  7.175175

1     0.0  0.592732  1.588421  7.586157   9.623011     0.0  6.816049

2     0.0  3.511759  4.327199  6.881787   9.870730     0.0  6.972130

3     0.0  3.663618  4.507649  6.659068  10.196184     0.0  7.843375

4     0.0  2.655741  2.821547  6.539454   9.738265     0.0  6.566967


     gene_7  gene_8  gene_9  ...  gene_20521  gene_20522
gene_20523  \
0  0.591871     0.0     0.0  ...    4.926711    8.210257    9.723516

1  0.000000     0.0     0.0  ...    4.593372    7.323865    9.740931

2  0.452595     0.0     0.0  ...    5.125213    8.127123   10.908640

3  0.434882     0.0     0.0  ...    6.076566    8.792959   10.141520

4  0.360982     0.0     0.0  ...    5.996032    8.891425   10.373790


   gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
0    7.220030    9.119813   12.003135    9.650743    8.921326
5.286759
1    6.256586    8.381612   12.674552   10.517059    9.397854
2.094168
2    5.401607    9.911597    9.045255    9.788359   10.090470
1.683023
3    8.942805    9.601208   11.392682    9.694814    9.684365
3.292001
4    7.181162    9.846910   11.922439    9.217749    9.461191
5.110372

   gene_20530
0         0.0
1         0.0
```

```
2         0.0
3         0.0
4         0.0

[5 rows x 20531 columns]
```

```
y = deep_learning_data['Class']
```

```
y.head(5)
```

```
0    PRAD
1    LUAD
2    PRAD
3    PRAD
4    BRCA
Name: Class, dtype: object
```

```
(trainX, testX, trainY, testY) = train_test_split(x, y,
test_size=0.20, random_state=42)
```

```
master_data.shape
```

```
(801, 20533)
```

```
deep_learning_data.shape
```

```
(801, 20533)
```

```
print(x.shape)
print(y.shape)
```

```
(801, 20531)
(801,)
```

```
trainX.head(5)
```

```
     gene_0    gene_1    gene_2    gene_3     gene_4  gene_5    gene_6
\
616     0.0  5.257467  3.658154  7.475920  10.532229     0.0  6.581443

329     0.0  3.972858  3.368908  6.644179   9.361722     0.0  8.306344

342     0.0  3.571592  3.900113  6.351773   9.740098     0.0  8.413133

394     0.0  3.835964  4.914249  6.419020   9.200234     0.0  7.591784

79      0.0  3.219029  2.284781  8.124178  10.386132     0.0  5.780310


       gene_7  gene_8  gene_9  ...  gene_20521  gene_20522  gene_20523
\
616  0.000000     0.0     0.0  ...    5.898494    9.431353   10.331006
```

```
329  0.431142     0.0     0.0  ...    6.431859    9.140965   10.624622

342  0.427284     0.0     0.0  ...    6.270166    9.507777    9.456317

394  0.629939     0.0     0.0  ...    6.871314    9.930158   10.783563

79   0.521252     0.0     0.0  ...    4.235850    8.706959    9.637483


     gene_20524  gene_20525  gene_20526  gene_20527  gene_20528
gene_20529  \
616    8.580372   10.036847   11.432338   10.221418   11.005996
5.364131
329    4.984161    9.985159   12.167415   10.983450    9.210472
10.333625
342    7.227991    9.542202   11.353886   10.655405    9.469707
8.015928
394    2.099363   10.293610   12.520019   11.092585    9.215594
3.764824
79     0.521252    9.059715   12.218963   10.892148    9.083054
3.460572

     gene_20530
616         0.0
329         0.0
342         0.0
394         0.0
79          0.0

[5 rows x 20531 columns]
```

```
trainY.head(5)
```

```
616    BRCA
329    BRCA
342    BRCA
394    LUAD
79     COAD
Name: Class, dtype: object
```

```
trainX.shape, testX.shape, trainY.shape, testY.shape
```

```
((640, 20531), (161, 20531), (640,), (161,))
```

```
from sklearn.preprocessing import LabelBinarizer
```

```
lb = LabelBinarizer()
```

```
trainY = lb.fit_transform(trainY)
testY = lb.transform(testY)

trainY

array([[1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       ...,
       [0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0],
       [1, 0, 0, 0, 0]])
```

param_number = output_channel_number * (input_channel_number * kernel_height * kernel_width + 1)

param_number = output_channel_number * (input_channel_number + 1) *italicized text*

```
import tensorflow
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV

def create_model(layers, activation):
    model = Sequential()
    for i, nodes in enumerate(layers):
        if i==0:
            model.add(Dense(nodes, input_dim=trainX.shape[1],
kernel_initializer="uniform", kernel_regularizer=l2(0.0002)))
            model.add(Activation(activation))
            model.add(BatchNormalization())
            model.add(Dropout(0.25))
        else:
            model.add(Dense(nodes, kernel_initializer="uniform",
kernel_regularizer=l2(0.0002)))
            model.add(Activation(activation))
            model.add(BatchNormalization())
            model.add(Dropout(0.25))

    # model.add(Dense(units = 1, kernel_initializer= 'glorot_uniform',
activation = 'sigmoid')) # Note: no activation beyond this point
    model.add(Dense(units=5, activation='softmax'))

    opt = Adam(lr=0.0001, beta_1=0.5, decay=0.0002 / 30)
    model.compile(loss="categorical_crossentropy", optimizer=opt,
metrics=["accuracy"])
    return model
```

```
model = KerasClassifier(build_fn=create_model, verbose=0)

model
```

```
<tensorflow.python.keras.wrappers.scikit_learn.KerasClassifier at
0x7f7795b888d0>
```

```
layers = [[20], [40, 20], [45, 30, 15], [50,35,20]]
activations = ['sigmoid', 'relu', 'elu']
param_grid = dict(layers=layers, activation=activations, batch_size =
[128, 256], epochs=[30])
grid = GridSearchCV(estimator=model, param_grid=param_grid,cv=5)

grid_result = grid.fit(trainX, trainY, validation_data=(testX, testY))
# , batch_size=32, validation_data=(testX, testY), epochs=[30])

print([grid_result.best_score_,grid_result.best_params_])
```

```
[0.9796875, {'activation': 'elu', 'batch_size': 128, 'epochs': 30,
'layers': [40, 20]}]
```

```
tensorflow.keras.backend.clear_session()

# define the architecture of the network
model = Sequential()
model.add(Dense(50, input_dim=trainX.shape[1],
kernel_initializer="uniform", kernel_regularizer=l2(0.0002),
name="LAYER____1"))
model.add(Activation("elu"))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Dense(35, kernel_initializer="uniform",
kernel_regularizer=l2(0.0002), name="LAYER____2"))
model.add(Activation("elu"))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Dense(20, kernel_initializer="uniform",
kernel_regularizer=l2(0.0002), name="LAYER____3"))
model.add(Activation("elu"))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Dense(5, activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
LAYER____1 (Dense)           (None, 50)                1026600
_________________________________________________________________
activation (Activation)      (None, 50)                0
```

```
_________________________________________________________________
batch_normalization (BatchNo (None, 50)                200
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
LAYER___2 (Dense)            (None, 35)                1785
_________________________________________________________________
activation_1 (Activation)    (None, 35)                0
_________________________________________________________________
batch_normalization_1 (Batch (None, 35)                140
_________________________________________________________________
dropout_1 (Dropout)          (None, 35)                0
_________________________________________________________________
LAYER___3 (Dense)            (None, 20)                720
_________________________________________________________________
activation_2 (Activation)    (None, 20)                0
_________________________________________________________________
batch_normalization_2 (Batch (None, 20)                80
_________________________________________________________________
dropout_2 (Dropout)          (None, 20)                0
_________________________________________________________________
dense (Dense)                (None, 5)                 105
=================================================================
Total params: 1,029,630
Trainable params: 1,029,420
Non-trainable params: 210
_________________________________________________________________
```

```
# ADAM OPTIMIZER
print("[INFO] compiling model...")

NUM_EPOCHS=50

opt = Adam(lr=0.0001, beta_1=0.5, decay=0.0002 / NUM_EPOCHS)
model.compile(loss="categorical_crossentropy", optimizer=opt,
metrics=["accuracy"])
H = model.fit(trainX, trainY, batch_size=128,
                        validation_data=(testX, testY),
epochs=NUM_EPOCHS, verbose=1)
```

```
[INFO] compiling model...
Epoch 1/50
5/5 [==============================] - 0s 96ms/step - loss: 1.9567 -
accuracy: 0.3203 - val_loss: 1.6712 - val_accuracy: 0.3416
Epoch 2/50
5/5 [==============================] - 0s 56ms/step - loss: 1.2434 -
accuracy: 0.5641 - val_loss: 1.6225 - val_accuracy: 0.4907
Epoch 3/50
5/5 [==============================] - 0s 55ms/step - loss: 0.9465 -
accuracy: 0.7625 - val_loss: 1.5990 - val_accuracy: 0.6460
```

```
Epoch 4/50
5/5 [==============================] - 0s 56ms/step - loss: 0.8657 -
accuracy: 0.7906 - val_loss: 1.5836 - val_accuracy: 0.8075
Epoch 5/50
5/5 [==============================] - 0s 60ms/step - loss: 0.7517 -
accuracy: 0.8438 - val_loss: 1.5698 - val_accuracy: 0.7950
Epoch 6/50
5/5 [==============================] - 0s 55ms/step - loss: 0.6853 -
accuracy: 0.8750 - val_loss: 1.5502 - val_accuracy: 0.8261
Epoch 7/50
5/5 [==============================] - 0s 57ms/step - loss: 0.6270 -
accuracy: 0.8984 - val_loss: 1.5496 - val_accuracy: 0.9193
Epoch 8/50
5/5 [==============================] - 0s 56ms/step - loss: 0.5892 -
accuracy: 0.9094 - val_loss: 1.5439 - val_accuracy: 0.9627
Epoch 9/50
5/5 [==============================] - 0s 54ms/step - loss: 0.5672 -
accuracy: 0.9453 - val_loss: 1.5291 - val_accuracy: 0.9689
Epoch 10/50
5/5 [==============================] - 0s 55ms/step - loss: 0.5386 -
accuracy: 0.9453 - val_loss: 1.5121 - val_accuracy: 0.9689
Epoch 11/50
5/5 [==============================] - 0s 57ms/step - loss: 0.5116 -
accuracy: 0.9500 - val_loss: 1.4943 - val_accuracy: 0.9814
Epoch 12/50
5/5 [==============================] - 0s 56ms/step - loss: 0.4809 -
accuracy: 0.9703 - val_loss: 1.4816 - val_accuracy: 0.9814
Epoch 13/50
5/5 [==============================] - 0s 53ms/step - loss: 0.4771 -
accuracy: 0.9609 - val_loss: 1.4639 - val_accuracy: 0.9876
Epoch 14/50
5/5 [==============================] - 0s 60ms/step - loss: 0.4637 -
accuracy: 0.9656 - val_loss: 1.4400 - val_accuracy: 0.9814
Epoch 15/50
5/5 [==============================] - 0s 55ms/step - loss: 0.4463 -
accuracy: 0.9750 - val_loss: 1.4197 - val_accuracy: 0.9876
Epoch 16/50
5/5 [==============================] - 0s 57ms/step - loss: 0.4236 -
accuracy: 0.9875 - val_loss: 1.4051 - val_accuracy: 0.9876
Epoch 17/50
5/5 [==============================] - 0s 59ms/step - loss: 0.4171 -
accuracy: 0.9906 - val_loss: 1.3886 - val_accuracy: 0.9876
Epoch 18/50
5/5 [==============================] - 0s 59ms/step - loss: 0.4073 -
accuracy: 0.9828 - val_loss: 1.3668 - val_accuracy: 0.9938
Epoch 19/50
5/5 [==============================] - 0s 54ms/step - loss: 0.3981 -
accuracy: 0.9828 - val_loss: 1.3436 - val_accuracy: 0.9938
Epoch 20/50
5/5 [==============================] - 0s 58ms/step - loss: 0.3838 -
```

```
accuracy: 0.9859 - val_loss: 1.3311 - val_accuracy: 1.0000
Epoch 21/50
5/5 [==============================] - 0s 78ms/step - loss: 0.3760 -
accuracy: 0.9937 - val_loss: 1.3119 - val_accuracy: 1.0000
Epoch 22/50
5/5 [==============================] - 0s 93ms/step - loss: 0.3685 -
accuracy: 0.9891 - val_loss: 1.2876 - val_accuracy: 1.0000
Epoch 23/50
5/5 [==============================] - 0s 91ms/step - loss: 0.3738 -
accuracy: 0.9859 - val_loss: 1.2584 - val_accuracy: 1.0000
Epoch 24/50
5/5 [==============================] - 0s 96ms/step - loss: 0.3604 -
accuracy: 0.9922 - val_loss: 1.2353 - val_accuracy: 1.0000
Epoch 25/50
5/5 [==============================] - 0s 93ms/step - loss: 0.3465 -
accuracy: 0.9953 - val_loss: 1.2153 - val_accuracy: 1.0000
Epoch 26/50
5/5 [==============================] - 0s 95ms/step - loss: 0.3491 -
accuracy: 0.9906 - val_loss: 1.1915 - val_accuracy: 1.0000
Epoch 27/50
5/5 [==============================] - 0s 93ms/step - loss: 0.3570 -
accuracy: 0.9922 - val_loss: 1.1771 - val_accuracy: 1.0000
Epoch 28/50
5/5 [==============================] - 0s 97ms/step - loss: 0.3452 -
accuracy: 0.9875 - val_loss: 1.1478 - val_accuracy: 0.9938
Epoch 29/50
5/5 [==============================] - 0s 95ms/step - loss: 0.3382 -
accuracy: 0.9922 - val_loss: 1.3652 - val_accuracy: 0.3354
Epoch 30/50
5/5 [==============================] - 0s 92ms/step - loss: 0.3279 -
accuracy: 0.9984 - val_loss: 1.2150 - val_accuracy: 0.5217
Epoch 31/50
5/5 [==============================] - 0s 91ms/step - loss: 0.3305 -
accuracy: 0.9922 - val_loss: 1.1506 - val_accuracy: 0.6894
Epoch 32/50
5/5 [==============================] - 0s 98ms/step - loss: 0.3324 -
accuracy: 0.9922 - val_loss: 1.0788 - val_accuracy: 0.9193
Epoch 33/50
5/5 [==============================] - 0s 81ms/step - loss: 0.3191 -
accuracy: 0.9953 - val_loss: 1.0502 - val_accuracy: 0.9565
Epoch 34/50
5/5 [==============================] - 0s 57ms/step - loss: 0.3181 -
accuracy: 0.9953 - val_loss: 1.0142 - val_accuracy: 0.9752
Epoch 35/50
5/5 [==============================] - 0s 55ms/step - loss: 0.3122 -
accuracy: 0.9984 - val_loss: 0.9635 - val_accuracy: 0.9876
Epoch 36/50
5/5 [==============================] - 0s 57ms/step - loss: 0.3241 -
accuracy: 0.9953 - val_loss: 0.9223 - val_accuracy: 1.0000
Epoch 37/50
```

```
5/5 [==============================] - 0s 56ms/step - loss: 0.3086 -
accuracy: 0.9953 - val_loss: 0.8907 - val_accuracy: 1.0000
Epoch 38/50
5/5 [==============================] - 0s 55ms/step - loss: 0.3112 -
accuracy: 0.9969 - val_loss: 0.8607 - val_accuracy: 1.0000
Epoch 39/50
5/5 [==============================] - 0s 55ms/step - loss: 0.3087 -
accuracy: 0.9953 - val_loss: 0.8310 - val_accuracy: 1.0000
Epoch 40/50
5/5 [==============================] - 0s 57ms/step - loss: 0.2968 -
accuracy: 0.9953 - val_loss: 0.8129 - val_accuracy: 1.0000
Epoch 41/50
5/5 [==============================] - 0s 54ms/step - loss: 0.3125 -
accuracy: 0.9937 - val_loss: 0.7897 - val_accuracy: 1.0000
Epoch 42/50
5/5 [==============================] - 0s 56ms/step - loss: 0.3046 -
accuracy: 0.9953 - val_loss: 0.7653 - val_accuracy: 1.0000
Epoch 43/50
5/5 [==============================] - 0s 60ms/step - loss: 0.2868 -
accuracy: 0.9969 - val_loss: 0.7342 - val_accuracy: 1.0000
Epoch 44/50
5/5 [==============================] - 0s 53ms/step - loss: 0.2968 -
accuracy: 0.9984 - val_loss: 0.7042 - val_accuracy: 1.0000
Epoch 45/50
5/5 [==============================] - 0s 56ms/step - loss: 0.3008 -
accuracy: 0.9984 - val_loss: 0.6788 - val_accuracy: 1.0000
Epoch 46/50
5/5 [==============================] - 0s 56ms/step - loss: 0.2818 -
accuracy: 1.0000 - val_loss: 0.6517 - val_accuracy: 1.0000
Epoch 47/50
5/5 [==============================] - 0s 55ms/step - loss: 0.2793 -
accuracy: 0.9984 - val_loss: 0.7952 - val_accuracy: 0.9876
Epoch 48/50
5/5 [==============================] - 0s 54ms/step - loss: 0.2830 -
accuracy: 0.9984 - val_loss: 0.6304 - val_accuracy: 1.0000
Epoch 49/50
5/5 [==============================] - 0s 56ms/step - loss: 0.2817 -
accuracy: 0.9953 - val_loss: 0.6144 - val_accuracy: 1.0000
Epoch 50/50
5/5 [==============================] - 0s 55ms/step - loss: 0.2899 -
accuracy: 0.9922 - val_loss: 0.5669 - val_accuracy: 1.0000
```

```
# save the network to disk
print("[INFO] serializing network...")
model.save("/content/cluster_weights.hdf5")
```

```
[INFO] serializing network...
```

```
from sklearn.metrics import classification_report

# evaluate the network
```

```
print("[INFO] evaluating network...")
predictions = model.predict(testX, batch_size=128)
print(classification_report(testY.argmax(axis=1),
predictions.argmax(axis=1),
target_names=[str(x) for x in lb.classes_]))
```

```
[INFO] evaluating network...
              precision    recall  f1-score   support

        BRCA       1.00      1.00      1.00        61
        COAD       1.00      1.00      1.00        17
        KIRC       1.00      1.00      1.00        25
        LUAD       1.00      1.00      1.00        29
        PRAD       1.00      1.00      1.00        29

    accuracy                           1.00       161
   macro avg       1.00      1.00      1.00       161
weighted avg       1.00      1.00      1.00       161
```

```
xyz = model.predict(testX)
y_pr=[]
for k in xyz:
    y_pr.append(np.argmax(k))

y_val=[]
for k in testY:
    y_val.append(np.argmax(k))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_val, y_pr)
```

```
array([[61,  0,  0,  0,  0],
       [ 0, 17,  0,  0,  0],
       [ 0,  0, 25,  0,  0],
       [ 0,  0,  0, 29,  0],
       [ 0,  0,  0,  0, 29]])
```

```
# plot the training loss and accuracy
plt.style.use("ggplot")
plt.figure()

plt.plot(np.arange(0, 50), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, 50), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, 50), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, 50), H.history["val_accuracy"], label="val_acc")

plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
```

```
plt.legend()
plt.savefig("/content/train_val_graph.jpg")
```