Skin Cancer Detection

Importing the required libraries

```
import os
import cv2
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
from skimage.io import imread
from skimage.transform import resize
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, roc_curve
```

Accessing the images from the directory: Benign, Malignant

```
DATADIR =
'/Users/dexter/Desktop/D-GALLOS/CODES/PYTHON/FIles/Skin_Cancer_Dataset
/DATA2'
CATEGORIES = ['benign','malignant']

# Reading, Resizing, Flattening, Gray
# Orignal image size was 225x225 - we have converted that into 200x200
in 2 - Dimension

label = []
train_data = []

def create_training_data():
    for category in CATEGORIES:
        class_num=CATEGORIES.index(category)
        path=os.path.join(DATADIR,category)
        for img in os.listdir(path):
            img_array=imread(os.path.join(path,img))
            img_resized=resize(img_array,(30,30,3))
            img_black = np.mean(img_resized, axis=2) #Standardising
the values too
            train_data.append(img_black.flatten())
```

```
            label.append(class_num)
create_training_data()
train_data=np.array(train_data)
label=np.array(label)
```

Exploratory Data Analysis

```
# This is our label array in correspondance with each image
# Showing the particular image came from which type
print(label)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```
# This is the lenght of our data variables/ Pixels
train_data
```

```
array([[0.77247785, 0.77254902, 0.76862745, ..., 0.73442266,
0.73594771,
```

```
        0.72586638],
       [0.72102977, 0.72600871, 0.73093682, ..., 0.70623094,
0.68849237,
        0.68204793],
       [0.63459695, 0.6188671 , 0.58239651, ..., 0.64095861,
0.58152505,
        0.44277705],
       ...,
       [0.49163399, 0.53233551, 0.54018155, ..., 0.40570806,
0.38505011,
        0.36538707],
       [0.47777778, 0.59681481, 0.66434277, ..., 0.55937545,
0.48208715,
        0.35429194],
       [0.71087727, 0.70959477, 0.70801743, ..., 0.68954248,
0.67572985,
        0.6698199 ]])
```

```
# This is our data in array format/ images in array format
x=pd.DataFrame(train_data)
y=pd.DataFrame(label)

x
```

```
          0         1         2         3         4         5
6    \
0    0.772478  0.772549  0.768627  0.769935  0.774584  0.781830
0.782025
1    0.721030  0.726009  0.730937  0.739278  0.747813  0.753508
0.729368
2    0.634597  0.618867  0.582397  0.575300  0.528092  0.605839
0.674379
3    0.702876  0.725621  0.750407  0.762054  0.762092  0.777952
0.779065
4    0.673436  0.687098  0.695643  0.712829  0.716632  0.704398
0.691537
..        ...       ...       ...       ...       ...       ...
...
655  0.378071  0.409490  0.440646  0.476296  0.502052  0.514466
0.513659
656  0.465506  0.489808  0.501394  0.507146  0.490763  0.506318
0.555633
657  0.491634  0.532336  0.540182  0.591373  0.631765  0.653640
0.682755
658  0.477778  0.596815  0.664343  0.688161  0.689965  0.682092
0.717147
659  0.710877  0.709595  0.708017  0.706032  0.700523  0.695425
0.704662

          7         8         9    ...       890       891
892  \
```

0	0.775861	0.775153	0.772244	...	0.744834	0.756273	0.754902
1	0.743442	0.761012	0.757910	...	0.736819	0.734641	0.707342
2	0.719063	0.735948	0.736555	...	0.751200	0.747692	0.734641
3	0.776471	0.780979	0.786318	...	0.751634	0.742484	0.735294
4	0.701656	0.708443	0.717647	...	0.689021	0.687277	0.687582
..	...	...	...	...	...	...	...
655	0.516950	0.525544	0.526228	...	0.538909	0.532549	0.520370
656	0.583159	0.580731	0.595425	...	0.583445	0.572428	0.527407
657	0.698845	0.723583	0.714728	...	0.494684	0.506667	0.489935
658	0.738519	0.738911	0.735380	...	0.681044	0.702702	0.706885
659	0.712113	0.702712	0.677907	...	0.741611	0.738562	0.735948

	893	894	895	896	897	898	899
0	0.737037	0.738694	0.741176	0.728677	0.734423	0.735948	0.725866
1	0.680687	0.666972	0.704257	0.717535	0.706231	0.688492	0.682048
2	0.713823	0.699085	0.689847	0.666057	0.640959	0.581525	0.442777
3	0.725795	0.723920	0.730444	0.724818	0.720370	0.713155	0.697037
4	0.687625	0.678999	0.677124	0.674360	0.665323	0.660436	0.657821
..	...	...	...	...	...	...	...
655	0.515043	0.517691	0.499638	0.473402	0.428453	0.374331	0.300317
656	0.518039	0.518125	0.490980	0.447059	0.447887	0.430105	0.399679
657	0.463553	0.407887	0.385621	0.394423	0.405708	0.385050	0.365387
658	0.694510	0.680652	0.667059	0.623728	0.559375	0.482087	0.354292
659	0.726831	0.720829	0.718824	0.707426	0.689542	0.675730	0.669820

[660 rows x 900 columns]

```
#Assigning th Independent and Dependent variables

dataset = x

dataset['label'] = y

dataset
```

```
            0         1         2         3         4         5
6  \
0    0.772478  0.772549  0.768627  0.769935  0.774584  0.781830
0.782025
1    0.721030  0.726009  0.730937  0.739278  0.747813  0.753508
0.729368
2    0.634597  0.618867  0.582397  0.575300  0.528092  0.605839
0.674379
3    0.702876  0.725621  0.750407  0.762054  0.762092  0.777952
0.779065
4    0.673436  0.687098  0.695643  0.712829  0.716632  0.704398
0.691537
..        ...       ...       ...       ...       ...       ...
...
655  0.378071  0.409490  0.440646  0.476296  0.502052  0.514466
0.513659
656  0.465506  0.489808  0.501394  0.507146  0.490763  0.506318
0.555633
657  0.491634  0.532336  0.540182  0.591373  0.631765  0.653640
0.682755
658  0.477778  0.596815  0.664343  0.688161  0.689965  0.682092
0.717147
659  0.710877  0.709595  0.708017  0.706032  0.700523  0.695425
0.704662

            7         8         9  ...       891       892
893  \
0    0.775861  0.775153  0.772244  ...  0.756273  0.754902  0.737037

1    0.743442  0.761012  0.757910  ...  0.734641  0.707342  0.680687

2    0.719063  0.735948  0.736555  ...  0.747692  0.734641  0.713823

3    0.776471  0.780979  0.786318  ...  0.742484  0.735294  0.725795

4    0.701656  0.708443  0.717647  ...  0.687277  0.687582  0.687625

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

655  0.516950  0.525544  0.526228  ...  0.532549  0.520370  0.515043
```

```
656  0.583159  0.580731  0.595425  ...  0.572428  0.527407  0.518039
657  0.698845  0.723583  0.714728  ...  0.506667  0.489935  0.463553
658  0.738519  0.738911  0.735380  ...  0.702702  0.706885  0.694510
659  0.712113  0.702712  0.677907  ...  0.738562  0.735948  0.726831

          894       895       896       897       898       899  label
0    0.738694  0.741176  0.728677  0.734423  0.735948  0.725866      0
1    0.666972  0.704257  0.717535  0.706231  0.688492  0.682048      0
2    0.699085  0.689847  0.666057  0.640959  0.581525  0.442777      0
3    0.723920  0.730444  0.724818  0.720370  0.713155  0.697037      0
4    0.678999  0.677124  0.674360  0.665323  0.660436  0.657821      0
..        ...       ...       ...       ...       ...       ...    ...
655  0.517691  0.499638  0.473402  0.428453  0.374331  0.300317      1
656  0.518125  0.490980  0.447059  0.447887  0.430105  0.399679      1
657  0.407887  0.385621  0.394423  0.405708  0.385050  0.365387      1
658  0.680652  0.667059  0.623728  0.559375  0.482087  0.354292      1
659  0.720829  0.718824  0.707426  0.689542  0.675730  0.669820      1

[660 rows x 901 columns]
```

Converting our dataset to CSV file

```
dataset.to_csv('dataset.csv',index=False)

data1 =
pd.read_csv('/Users/dexter/Desktop/D-GALLOS/CODES/PYTHON/Jupyter/Skin-
Cancer-Project/dataset.csv')

data1

            0         1         2         3         4         5
6  \
0    0.772478  0.772549  0.768627  0.769935  0.774584  0.781830
0.782025
```

```
1    0.721030  0.726009  0.730937  0.739278  0.747813  0.753508
0.729368
2    0.634597  0.618867  0.582397  0.575300  0.528092  0.605839
0.674379
3    0.702876  0.725621  0.750407  0.762054  0.762092  0.777952
0.779065
4    0.673436  0.687098  0.695643  0.712829  0.716632  0.704398
0.691537
..        ...       ...       ...       ...       ...       ...
...
655  0.378071  0.409490  0.440646  0.476296  0.502052  0.514466
0.513659
656  0.465506  0.489808  0.501394  0.507146  0.490763  0.506318
0.555633
657  0.491634  0.532336  0.540182  0.591373  0.631765  0.653640
0.682755
658  0.477778  0.596815  0.664343  0.688161  0.689965  0.682092
0.717147
659  0.710877  0.709595  0.708017  0.706032  0.700523  0.695425
0.704662

            7         8         9  ...       891       892
893  \
0    0.775861  0.775153  0.772244  ...  0.756273  0.754902  0.737037

1    0.743442  0.761012  0.757910  ...  0.734641  0.707342  0.680687

2    0.719063  0.735948  0.736555  ...  0.747692  0.734641  0.713823

3    0.776471  0.780979  0.786318  ...  0.742484  0.735294  0.725795

4    0.701656  0.708443  0.717647  ...  0.687277  0.687582  0.687625

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

655  0.516950  0.525544  0.526228  ...  0.532549  0.520370  0.515043

656  0.583159  0.580731  0.595425  ...  0.572428  0.527407  0.518039

657  0.698845  0.723583  0.714728  ...  0.506667  0.489935  0.463553

658  0.738519  0.738911  0.735380  ...  0.702702  0.706885  0.694510

659  0.712113  0.702712  0.677907  ...  0.738562  0.735948  0.726831


          894       895       896       897       898       899  label

0    0.738694  0.741176  0.728677  0.734423  0.735948  0.725866      0
```

```
1    0.666972  0.704257  0.717535  0.706231  0.688492  0.682048      0

2    0.699085  0.689847  0.666057  0.640959  0.581525  0.442777      0

3    0.723920  0.730444  0.724818  0.720370  0.713155  0.697037      0

4    0.678999  0.677124  0.674360  0.665323  0.660436  0.657821      0

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

655  0.517691  0.499638  0.473402  0.428453  0.374331  0.300317      1

656  0.518125  0.490980  0.447059  0.447887  0.430105  0.399679      1

657  0.407887  0.385621  0.394423  0.405708  0.385050  0.365387      1

658  0.680652  0.667059  0.623728  0.559375  0.482087  0.354292      1

659  0.720829  0.718824  0.707426  0.689542  0.675730  0.669820      1


[660 rows x 901 columns]
```

```
data2 = data1.drop(labels=['label'], axis=1)
data2.info
```

```
<bound method DataFrame.info of             0         1         2
3         4         5         6  \
0    0.772478  0.772549  0.768627  0.769935  0.774584  0.781830
0.782025
1    0.721030  0.726009  0.730937  0.739278  0.747813  0.753508
0.729368
2    0.634597  0.618867  0.582397  0.575300  0.528092  0.605839
0.674379
3    0.702876  0.725621  0.750407  0.762054  0.762092  0.777952
0.779065
4    0.673436  0.687098  0.695643  0.712829  0.716632  0.704398
0.691537
..        ...       ...       ...       ...       ...       ...
...
655  0.378071  0.409490  0.440646  0.476296  0.502052  0.514466
0.513659
656  0.465506  0.489808  0.501394  0.507146  0.490763  0.506318
0.555633
657  0.491634  0.532336  0.540182  0.591373  0.631765  0.653640
0.682755
658  0.477778  0.596815  0.664343  0.688161  0.689965  0.682092
0.717147
```

```
659  0.710877  0.709595  0.708017  0.706032  0.700523  0.695425
0.704662

          7         8         9  ...       890       891
892  \
0    0.775861  0.775153  0.772244  ...  0.744834  0.756273  0.754902

1    0.743442  0.761012  0.757910  ...  0.736819  0.734641  0.707342

2    0.719063  0.735948  0.736555  ...  0.751200  0.747692  0.734641

3    0.776471  0.780979  0.786318  ...  0.751634  0.742484  0.735294

4    0.701656  0.708443  0.717647  ...  0.689021  0.687277  0.687582

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

655  0.516950  0.525544  0.526228  ...  0.538909  0.532549  0.520370

656  0.583159  0.580731  0.595425  ...  0.583445  0.572428  0.527407

657  0.698845  0.723583  0.714728  ...  0.494684  0.506667  0.489935

658  0.738519  0.738911  0.735380  ...  0.681044  0.702702  0.706885

659  0.712113  0.702712  0.677907  ...  0.741611  0.738562  0.735948


          893       894       895       896       897       898
899
0    0.737037  0.738694  0.741176  0.728677  0.734423  0.735948
0.725866
1    0.680687  0.666972  0.704257  0.717535  0.706231  0.688492
0.682048
2    0.713823  0.699085  0.689847  0.666057  0.640959  0.581525
0.442777
3    0.725795  0.723920  0.730444  0.724818  0.720370  0.713155
0.697037
4    0.687625  0.678999  0.677124  0.674360  0.665323  0.660436
0.657821
..        ...       ...       ...       ...       ...       ...
...
655  0.515043  0.517691  0.499638  0.473402  0.428453  0.374331
0.300317
656  0.518039  0.518125  0.490980  0.447059  0.447887  0.430105
0.399679
657  0.463553  0.407887  0.385621  0.394423  0.405708  0.385050
0.365387
658  0.694510  0.680652  0.667059  0.623728  0.559375  0.482087
```

```
0.354292
659  0.726831  0.720829  0.718824  0.707426  0.689542  0.675730
0.669820

[660 rows x 900 columns]>
```

Converting our array data into Numpy array

```
data_np = np.array(data2)

# Displaying the Numpy Array data
print(data_np[0])

# Displaying the Rows of our Numpy Array
print("\nDimension of Numpy Array - ",data_np[0].shape)
```

```
[0.77247785 0.77254902 0.76862745 0.76993464 0.77458388 0.78183007
 0.78202469 0.77586057 0.77515323 0.77224401 0.76583878 0.75335221
 0.7330719  0.70910675 0.6936703  0.6827451  0.66814379 0.60819172
 0.64560203 0.6837342  0.71476979 0.73550182 0.76167756 0.76830211
 0.76862745 0.7795817  0.77986347 0.78045025 0.78793028 0.77570225
 0.77385621 0.77254902 0.76923747 0.76379085 0.76509804 0.76379085
 0.76640523 0.76816993 0.76732026 0.76413943 0.75437908 0.73031373
 0.68947712 0.65346405 0.64118083 0.64762963 0.65175163 0.60995643
 0.60286275 0.62810458 0.66901961 0.71154684 0.73169935 0.75010893
 0.75895861 0.76618301 0.76287582 0.76947712 0.77777778 0.76069281
 0.77302106 0.77254902 0.76412491 0.76801017 0.76472767 0.76670298
 0.76339869 0.76535948 0.77119826 0.76357298 0.74294118 0.70275236
 0.63809005 0.61261438 0.6163907  0.62323166 0.61810458 0.60094408
 0.59511983 0.60599129 0.61141612 0.62792302 0.67712418 0.73381264
 0.75151053 0.7540305  0.75603486 0.76670298 0.77124183 0.75716776
 0.77291649 0.76736383 0.76601307 0.77124183 0.77703268 0.77526071
 0.76732026 0.76577342 0.76421496 0.74979375 0.7220915  0.65921714
 0.58047204 0.54705882 0.55444735 0.56849092 0.53556863 0.54318083
 0.57664633 0.56701089 0.54381409 0.55015251 0.59895425 0.68526071
 0.72854031 0.74195643 0.75912854 0.76601307 0.76682353 0.74974147
 0.76296296 0.75141176 0.76067538 0.76378214 0.76732026 0.76508497
 0.7580305  0.7572549  0.75807407 0.73211329 0.6930719  0.6227451
 0.53311547 0.50601307 0.50265795 0.50015686 0.45150327 0.45941176
 0.53294553 0.52589542 0.51028322 0.51189542 0.5424183  0.63315033
 0.68750763 0.72222222 0.74718519 0.76302832 0.76261438 0.74801743
 0.75890922 0.74497168 0.75027596 0.75971532 0.76862745 0.76159622
 0.74949891 0.74668845 0.73569208 0.70097458 0.66830501 0.59904139
 0.49852578 0.47046187 0.50570661 0.45882353 0.39633987 0.4199419
 0.50693827 0.48458824 0.46204793 0.47813217 0.49265795 0.56631808
 0.65706463 0.71054466 0.73616558 0.75366739 0.75664052 0.74483515
 0.74893246 0.73764706 0.73877996 0.75067538 0.75943355 0.74956137
 0.73590269 0.72405229 0.68749601 0.6488642  0.63803922 0.5609862
 0.45720407 0.43464052 0.48417284 0.44818155 0.41378649 0.47164125
 0.51733043 0.4443573  0.39455919 0.43660276 0.46283224 0.52226725
 0.62923166 0.69995207 0.73037037 0.7463907  0.75294118 0.74640523
```

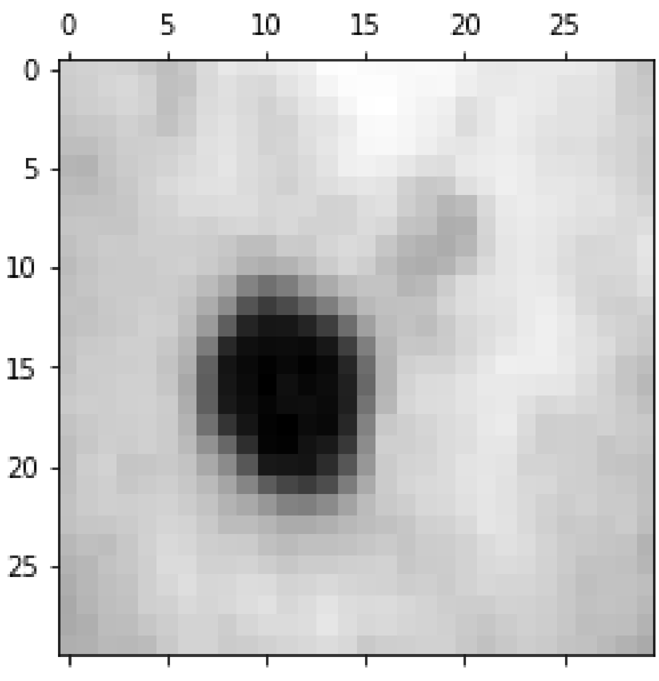
0.74328976	0.74045752	0.74379085	0.74771242	0.75424837	0.74566449
0.7208061	0.68627451	0.63102397	0.59533769	0.61496732	0.5596732
0.45599129	0.4227451	0.46313725	0.47627451	0.5051634	0.54825708
0.50740741	0.40444444	0.37291939	0.41464052	0.48267974	0.4979085
0.60867102	0.6927451	0.72448802	0.73311547	0.74333333	0.74901961
0.74379085	0.74117647	0.74505447	0.74463471	0.74775163	0.73522585
0.71355265	0.66276688	0.60287582	0.57389978	0.59358606	0.54724909
0.44840232	0.4431939	0.50123021	0.51250545	0.56608715	0.57408134
0.47558315	0.38117647	0.35980102	0.36993609	0.41784314	0.46392157
0.59442266	0.68148584	0.71472767	0.72897603	0.73813943	0.74122004
0.74178649	0.74509804	0.74509804	0.74379085	0.74237037	0.72810458
0.70588235	0.65376906	0.59522585	0.55124183	0.57204357	0.54745098
0.49036311	0.52401743	0.5725069	0.59616412	0.61511983	0.59195352
0.49336674	0.38776906	0.36699927	0.3751634	0.39429194	0.43885403
0.56881772	0.66187364	0.71228758	0.73620189	0.74118083	0.72841104
0.74107625	0.73986928	0.74117647	0.74248366	0.73070588	0.70791721
0.65960349	0.60633987	0.5592244	0.50409586	0.51699346	0.53746841
0.53847495	0.59783007	0.62662309	0.6383573	0.64160784	0.61283224
0.52148584	0.42494118	0.40894989	0.39411765	0.39705882	0.44126362
0.55109804	0.65254902	0.71089325	0.741939	0.74858824	0.72971678
0.74200436	0.73488889	0.73938998	0.73856209	0.72283224	0.67861583
0.60075236	0.52498911	0.5071024	0.48175309	0.478122	0.53084967
0.56423384	0.62040523	0.64910094	0.64948003	0.64793028	0.62884532
0.55816848	0.49664488	0.48424982	0.45021206	0.43583878	0.47039797
0.56048947	0.66300654	0.71154539	0.74357298	0.74227887	0.73475236
0.73333333	0.73318083	0.73482208	0.72783588	0.698061	0.64383442
0.56008715	0.46862745	0.45979666	0.47991285	0.48823529	0.51700073
0.57643428	0.62666667	0.64702251	0.65615832	0.65228758	0.64705882
0.58948439	0.56339869	0.53507625	0.47647059	0.43366013	0.48305737
0.58400145	0.66324619	0.71473493	0.73834423	0.73782135	0.74618736
0.73594771	0.7356732	0.73474946	0.72188671	0.66937255	0.5977342
0.50296296	0.42738562	0.4144488	0.46479303	0.48849673	0.50583878
0.5803268	0.62954248	0.64061002	0.65027887	0.65386928	0.64932462
0.62187364	0.59973856	0.56287582	0.50466231	0.43901961	0.5064183
0.62230501	0.68013072	0.72659259	0.73986928	0.73986928	0.74993028
0.73233115	0.73534205	0.73206245	0.69797095	0.6124488	0.53028177
0.4588032	0.41176471	0.41215686	0.42845461	0.45384314	0.50481772
0.57694263	0.62471024	0.63844299	0.65159041	0.65258824	0.63249818
0.60958606	0.58362963	0.54941322	0.508061	0.47605664	0.53092665
0.63476979	0.69490196	0.73037037	0.73037037	0.73934641	0.7503268
0.73240232	0.73734205	0.72867102	0.68588235	0.57549891	0.4788671
0.42143791	0.41045752	0.43525054	0.41464198	0.46039216	0.52470588
0.58578068	0.61899346	0.62261438	0.63113871	0.64705882	0.62753813
0.60553377	0.57623965	0.54753522	0.52287582	0.51514161	0.56461874
0.6334626	0.69517647	0.73056354	0.7245679	0.74487582	0.75247785
0.72728105	0.72469281	0.72135076	0.68458824	0.59433987	0.49294553
0.42663617	0.40895425	0.44052288	0.45145534	0.47816993	0.53920261
0.58189542	0.59150327	0.58165142	0.60087582	0.62248366	0.62108932
0.59568627	0.58681046	0.54462309	0.50248366	0.50176471	0.57694989
0.641878	0.70479739	0.73497603	0.72172113	0.75515033	0.75978214

0.72915759	0.72156863	0.72374728	0.69163399	0.65736383	0.5717284
0.49687001	0.45555556	0.48570806	0.50078431	0.49455338	0.52905592
0.53420479	0.52686275	0.5362963	0.55821351	0.58505447	0.586565
0.58030501	0.59045752	0.51904866	0.44954975	0.47265795	0.57355846
0.65538126	0.70668845	0.73904139	0.7385984	0.75941176	0.74941176
0.73202614	0.72820479	0.72099492	0.69054757	0.6724183	0.60416267
0.5417589	0.49061002	0.49496151	0.50862745	0.50811765	0.52989107
0.51381264	0.49318954	0.51274219	0.5056398	0.53953377	0.53854031
0.57093537	0.5854902	0.53028322	0.44669426	0.48429194	0.61098039
0.67914887	0.71808279	0.74409441	0.75315178	0.75041394	0.7404793
0.72958606	0.73333333	0.72248366	0.7031329	0.67111111	0.59960784
0.54492375	0.49130719	0.45699782	0.48671024	0.51973856	0.52605664
0.48298475	0.46762092	0.44321133	0.43288453	0.47359477	0.52108932
0.57450545	0.59073203	0.5714902	0.55542919	0.56764706	0.64558606
0.70727669	0.72783007	0.74979956	0.76056645	0.74962092	0.73119826
0.7254902	0.73242266	0.7291939	0.71824691	0.67918519	0.61893827
0.54685984	0.46771242	0.43642702	0.45437908	0.50257081	0.50213508
0.48222222	0.46514597	0.3891939	0.40230792	0.44514161	0.48801743
0.56091503	0.59359477	0.59420479	0.59367611	0.60185185	0.67302832
0.72058243	0.72784314	0.74578504	0.75481481	0.74566449	0.72924038
0.72553377	0.72849673	0.72915759	0.72122004	0.69233115	0.6351634
0.54993755	0.46479303	0.42614379	0.45472767	0.48972549	0.46503849
0.46649237	0.45239216	0.39472767	0.41599129	0.44807407	0.45855483
0.5362963	0.58972549	0.60938707	0.62122004	0.64466231	0.69133188
0.72992883	0.73864488	0.74252723	0.74639797	0.73944662	0.72606681
0.72795207	0.72372549	0.71840959	0.72287582	0.70333333	0.64943355
0.57616558	0.49379085	0.43141612	0.45640523	0.44934641	0.4243573
0.42930283	0.43921569	0.4154902	0.44492375	0.49346405	0.46895425
0.52492375	0.58228758	0.62620915	0.6628976	0.68823529	0.71562092
0.72671024	0.74150327	0.75163399	0.75108932	0.73542484	0.72679739
0.72152505	0.71764706	0.7110748	0.72156863	0.71198257	0.67074365
0.60209296	0.51230937	0.46775454	0.4780581	0.39947277	0.40410748
0.43931009	0.45699346	0.45287436	0.50285548	0.53271024	0.51896151
0.55378504	0.60427451	0.65041975	0.68762672	0.71427015	0.73198112
0.73649528	0.73965577	0.74855628	0.75323893	0.73424837	0.73041394
0.72287582	0.72048366	0.72139434	0.7239114	0.71846187	0.68207553
0.62092084	0.54605664	0.54646768	0.53050109	0.41661438	0.40858388
0.47519971	0.49735076	0.48605374	0.53977923	0.56253159	0.56312273
0.59235439	0.6377037	0.68701525	0.7130472	0.72895425	0.7409833
0.74248366	0.74366013	0.74866086	0.75294118	0.73538562	0.73494553
0.71764706	0.7220915	0.72788671	0.7242658	0.7190719	0.69172113
0.63595207	0.58124183	0.5916732	0.57683224	0.47503268	0.47385621
0.53041394	0.54465359	0.5464488	0.5730719	0.60043137	0.61906318
0.63637908	0.6745098	0.70397821	0.72410022	0.73915033	0.74118083
0.73912854	0.74484967	0.74819172	0.7503268	0.73546405	0.7356427
0.71497023	0.72684096	0.72257807	0.7124183	0.70793464	0.69373566
0.65918373	0.61849673	0.62188962	0.59111111	0.52192157	0.53472912
0.56269426	0.56972985	0.57784023	0.61440378	0.64591285	0.66720407
0.67986928	0.70182571	0.71018591	0.72601307	0.74034858	0.73856209
0.73484822	0.74457081	0.74331155	0.74592593	0.73710675	0.73464052

```
 0.70980392 0.71816993 0.71633987 0.70972404 0.70980392 0.70109659
 0.67437182 0.65163399 0.64936093 0.60649964 0.56625272 0.58949165
 0.59281046 0.60490196 0.62670298 0.65803195 0.68407407 0.70265069
 0.71411765 0.7177342  0.72068991 0.73581699 0.74771242 0.73991285
 0.74339869 0.7477342  0.74379085 0.73921569 0.73503268 0.73405955
 0.71333333 0.72026144 0.72281046 0.71323312 0.71159477 0.70740741
 0.68268845 0.68202614 0.66631808 0.63642702 0.60679739 0.62344227
 0.62379085 0.65134641 0.67243137 0.69428758 0.70861438 0.72217865
 0.72239651 0.7275817  0.73254031 0.74904575 0.75052288 0.73383007
 0.74356863 0.74505882 0.73899782 0.73287582 0.7372549  0.72902397
 0.71218446 0.72357734 0.72413217 0.71389978 0.71864924 0.71180973
 0.69507625 0.6975817  0.68344227 0.6621772  0.64122004 0.66152505
 0.66684096 0.69589978 0.70827887 0.71995643 0.7254902  0.73180828
 0.72612491 0.73735512 0.7448337  0.75627306 0.75490196 0.73703704
 0.73869426 0.74117647 0.72867683 0.73442266 0.73594771 0.72586638]

Dimension of Numpy Array -  (900,)
```

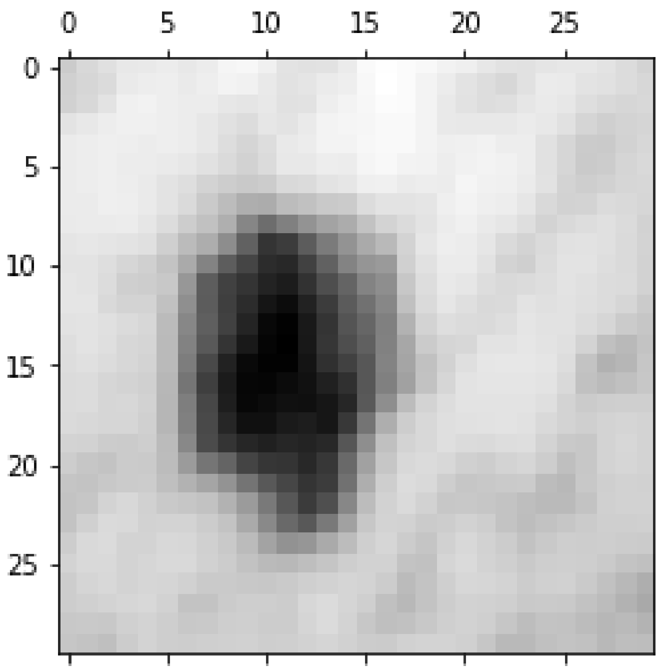
```
# This function sets up the matplotlib to work interactively.
# It lets you activate the matplotlib interactive support anywhere in an IPython session
%matplotlib inline
plt.gray()

# Here is the 300th image
plt.matshow(data_np[300].reshape(30,30))
```

```
<matplotlib.image.AxesImage at 0x7fdae0b41ee0>

<Figure size 432x288 with 0 Axes>
```



```
# Here is the 90th image
plt.matshow(data_np[90].reshape(30,30))
```

```
<matplotlib.image.AxesImage at 0x7fdad9d59b50>
```



Converting Numpy array to Dataframe

```
df = pd.DataFrame(data_np)
```

```
# Here is the first 5 rows of our Dataframe
df.head()
```

```
          0         1         2         3         4         5         6  \
0  0.772478  0.772549  0.768627  0.769935  0.774584  0.781830  0.782025
1  0.721030  0.726009  0.730937  0.739278  0.747813  0.753508  0.729368
2  0.634597  0.618867  0.582397  0.575300  0.528092  0.605839  0.674379
3  0.702876  0.725621  0.750407  0.762054  0.762092  0.777952  0.779065
4  0.673436  0.687098  0.695643  0.712829  0.716632  0.704398  0.691537

          7         8         9  ...       890       891       892       893  \
0  0.775861  0.775153  0.772244  ...  0.744834  0.756273  0.754902  0.737037
1  0.743442  0.761012  0.757910  ...  0.736819  0.734641  0.707342  0.680687
```

```
2  0.719063  0.735948  0.736555  ...  0.751200  0.747692  0.734641
0.713823
3  0.776471  0.780979  0.786318  ...  0.751634  0.742484  0.735294
0.725795
4  0.701656  0.708443  0.717647  ...  0.689021  0.687277  0.687582
0.687625

        894       895       896       897       898       899
0  0.738694  0.741176  0.728677  0.734423  0.735948  0.725866
1  0.666972  0.704257  0.717535  0.706231  0.688492  0.682048
2  0.699085  0.689847  0.666057  0.640959  0.581525  0.442777
3  0.723920  0.730444  0.724818  0.720370  0.713155  0.697037
4  0.678999  0.677124  0.674360  0.665323  0.660436  0.657821

[5 rows x 900 columns]
```

```
# Here is the Description of our Dataframe
df.describe()
```

```
              0           1           2           3           4
5    \
count  660.000000  660.000000  660.000000  660.000000  660.000000
660.000000
mean     0.583974    0.609440    0.627918    0.640136    0.650647
0.659770
std      0.187072    0.165919    0.150103    0.139049    0.130389
0.122166
min      0.000000    0.000000    0.000051    0.013072    0.022353
0.030372
25%      0.508392    0.535505    0.553252    0.570302    0.583173
0.596698
50%      0.631751    0.642022    0.654375    0.663031    0.668336
0.670238
75%      0.705922    0.719706    0.726607    0.730588    0.735644
0.738334
max      0.907190    0.920170    0.921569    0.924799    0.914566
0.907757

              6           7           8           9    ...         890
\
count  660.000000  660.000000  660.000000  660.000000  ...  660.000000

mean     0.665977    0.670592    0.673521    0.675621  ...    0.651999

std      0.118552    0.117089    0.116335    0.114404  ...    0.114909

min      0.032985    0.032680    0.030055    0.030198  ...    0.000000

25%      0.598396    0.603611    0.611237    0.614172  ...    0.587286
```

```
50%      0.677386    0.680185    0.683459    0.689172  ...    0.661853

75%      0.741261    0.746068    0.748995    0.750871  ...    0.722952

max      0.911493    0.912070    0.921438    0.955513  ...    0.971458


              891         892         893         894         895
896  \
count  660.000000  660.000000  660.000000  660.000000  660.000000
660.000000
mean     0.650726    0.649848    0.645033    0.638385    0.628263
0.616361
std      0.114456    0.113764    0.115582    0.119100    0.126432
0.138850
min      0.000000    0.000000    0.000000    0.000000    0.000000
0.000000
25%      0.588009    0.589847    0.580679    0.569227    0.559235
0.546866
50%      0.658845    0.659303    0.655408    0.651853    0.645556
0.639869
75%      0.720640    0.724913    0.719234    0.715249    0.708150
0.703767
max      0.961535    0.959020    0.915666    0.927409    0.972488
0.963187

              897         898         899
count  660.000000  660.000000  660.000000
mean     0.599699    0.579486    0.554473
std      0.158247    0.177457    0.196851
min      0.000000    0.000000    0.000000
25%      0.530067    0.509085    0.479566
50%      0.628758    0.620046    0.611174
75%      0.703322    0.692504    0.684794
max      0.959818    0.920784    0.901656

[8 rows x 900 columns]
```

```
# Here is information about rows and columns in our Dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 660 entries, 0 to 659
Columns: 900 entries, 0 to 899
dtypes: float64(900)
memory usage: 4.5 MB
```

```
df.mean(axis=0)
```

```
0      0.583974
1      0.609440
2      0.627918
3      0.640136
4      0.650647
         ...
895    0.628263
896    0.616361
897    0.599699
898    0.579486
899    0.554473
Length: 900, dtype: float64
```

```
df.std(axis=0)
```

```
0      0.187072
1      0.165919
2      0.150103
3      0.139049
4      0.130389
         ...
895    0.126432
896    0.138850
897    0.158247
898    0.177457
899    0.196851
Length: 900, dtype: float64
```

Normalising our Dataframe

Normalize Pixel Values For most image data, the pixel values are integers with values between 0 and 255.

Neural networks process inputs using small weight values, and inputs with large integer values can disrupt or slow down the learning process. As such it is good practice to normalize the pixel values so that each pixel value has a value between 0 and 1.

It is valid for images to have pixel values in the range 0-1 and images can be viewed normally.

This can be achieved by dividing all pixels values by the largest pixel value; that is 255. This is performed across all channels, regardless of the actual range of pixel values that are present in the image.

```
norm_df=(df-df.min())/(df.max()-df.min())
```

```
# This is normalised dataframe with values ranging between 0 and 1
norm_df
```

```
          0         1         2         3         4         5
6    \
0    0.851507  0.839572  0.834033  0.830142  0.843106  0.856474
```

```
0.852627
1    0.794795  0.788994  0.793133  0.796517  0.813101  0.824194
0.792688
2    0.699520  0.672557  0.631942  0.616663  0.566836  0.655888
0.730095
3    0.774784  0.788573  0.814261  0.821498  0.829105  0.852054
0.849258
4    0.742333  0.746708  0.754833  0.767508  0.778153  0.768221
0.749625
..        ...       ...       ...       ...       ...       ...
...
655  0.416750  0.445016  0.478119  0.508074  0.537651  0.551746
0.547148
656  0.513130  0.532302  0.544041  0.541910  0.524997  0.542460
0.594926
657  0.541931  0.578519  0.586132  0.634292  0.683034  0.710369
0.739629
658  0.526657  0.648592  0.720867  0.740451  0.748265  0.742797
0.778778
659  0.783604  0.771156  0.768261  0.760052  0.760098  0.757994
0.764566

          7         8         9    ...       890       891
892  \
0    0.845110  0.835890  0.801939  ...  0.766717  0.786527  0.787160

1    0.808245  0.820026  0.786448  ...  0.758467  0.764029  0.737568

2    0.780522  0.791907  0.763369  ...  0.773270  0.777602  0.766033

3    0.845803  0.842426  0.817149  ...  0.773717  0.772186  0.766714

4    0.760727  0.761051  0.742935  ...  0.709265  0.714770  0.716963

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

655  0.550689  0.555865  0.536066  ...  0.554743  0.553853  0.542607

656  0.625979  0.617777  0.610848  ...  0.600587  0.595328  0.549944

657  0.757531  0.778036  0.739780  ...  0.509218  0.526935  0.510870

658  0.802646  0.795231  0.762099  ...  0.701054  0.730812  0.737091

659  0.772619  0.754621  0.699987  ...  0.763400  0.768107  0.767396


          893       894       895       896       897       898
899
```

```
0    0.804919  0.796514  0.762145  0.756527  0.765168  0.799262
0.805037
1    0.743379  0.719178  0.724181  0.744960  0.735796  0.747724
0.756439
2    0.779567  0.753804  0.709363  0.691514  0.667792  0.631554
0.491071
3    0.792642  0.780584  0.751109  0.752521  0.750528  0.774508
0.773063
4    0.750956  0.732147  0.696280  0.700135  0.693176  0.717253
0.729570
..        ...       ...       ...       ...       ...       ...
...
655  0.562479  0.558212  0.513773  0.491495  0.446390  0.406535
0.333072
656  0.565751  0.558680  0.504870  0.464146  0.466637  0.467107
0.443272
657  0.506246  0.439813  0.396530  0.409498  0.422693  0.418176
0.405240
658  0.758475  0.733929  0.685930  0.647568  0.582793  0.523561
0.392935
659  0.793773  0.777251  0.739159  0.734464  0.718409  0.733863
0.742878

[660 rows x 900 columns]
```

```
# Description of our normalised dataframe
norm_df.describe()
```

```
              0           1           2           3           4
5    \
count  660.000000  660.000000  660.000000  660.000000  660.000000
660.000000
mean     0.643717    0.662313    0.681340    0.687776    0.704197
0.717356
std      0.206211    0.180313    0.162887    0.152511    0.146142
0.139239
min      0.000000    0.000000    0.000000    0.000000    0.000000
0.000000
25%      0.560403    0.581964    0.600315    0.611181    0.628572
0.645470
50%      0.696382    0.697721    0.710051    0.712888    0.724022
0.729288
75%      0.778142    0.782145    0.788434    0.786986    0.799462
0.806899
max      1.000000    1.000000    1.000000    1.000000    1.000000
1.000000

              6           7           8           9  ...         890
\
count  660.000000  660.000000  660.000000  660.000000  ...  660.000000
```

```
mean     0.720531    0.725403    0.721873    0.697517  ...    0.671155

std      0.134946    0.133148    0.130511    0.123638  ...    0.118285

min      0.000000    0.000000    0.000000    0.000000  ...    0.000000

25%      0.643604    0.649236    0.652000    0.631109  ...    0.604541

50%      0.733517    0.736312    0.733023    0.712162  ...    0.681298

75%      0.806226    0.811230    0.806545    0.778841  ...    0.744193

max      1.000000    1.000000    1.000000    1.000000  ...    1.000000


              891         892         893         894         895
896  \
count  660.000000  660.000000  660.000000  660.000000  660.000000
660.000000
mean     0.676758    0.677617    0.704441    0.688354    0.646037
0.639918
std      0.119035    0.118626    0.126228    0.128423    0.130009
0.144157
min      0.000000    0.000000    0.000000    0.000000    0.000000
0.000000
25%      0.611532    0.615053    0.634160    0.613782    0.575056
0.567767
50%      0.685201    0.687476    0.715772    0.702875    0.663819
0.664325
75%      0.749468    0.755890    0.785477    0.771234    0.728184
0.730666
max      1.000000    1.000000    1.000000    1.000000    1.000000
1.000000

              897         898         899
count  660.000000  660.000000  660.000000
mean     0.624804    0.629340    0.614950
std      0.164872    0.192723    0.218321
min      0.000000    0.000000    0.000000
25%      0.552258    0.552882    0.531873
50%      0.655080    0.673389    0.677835
75%      0.732766    0.752081    0.759485
max      1.000000    1.000000    1.000000

[8 rows x 900 columns]
```

```python
# Mean for Normalised Data
norm_df.mean(axis=0)
```

```
0      0.643717
1      0.662313
2      0.681340
3      0.687776
4      0.704197
         ...
895    0.646037
896    0.639918
897    0.624804
898    0.629340
899    0.614950
Length: 900, dtype: float64
```

```python
# Standard Deviation for Normalised Data
norm_df.std(axis=0)
```

```
0      0.206211
1      0.180313
2      0.162887
3      0.152511
4      0.146142
         ...
895    0.130009
896    0.144157
897    0.164872
898    0.192723
899    0.218321
Length: 900, dtype: float64
```

Assigning values to our Independent and Dependent Variables, i.e. X, Y respectively

```python
X = norm_df
y = label
```

Preprocessing

One of the reasons that it's easy to get confused between scaling and normalization is because the terms are sometimes used interchangeably and, to make it even more confusing, they are very similar! In both cases, you're transforming the values of numeric variables so that the transformed data points have specific helpful properties. The difference is that, in scaling, you're changing the range of your data while in normalization you're changing the shape of the distribution of your data

Scaling our Independant Variable - X

```python
# The preprocessing module provides the StandardScaler utility class,
# which is a quick and easy way to perform the following operation on
an array-like dataset:

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

```
array([[ 1.00841848,  0.98380718,  0.93812951, ...,  0.85199723,
         0.88235805,  0.87133739],
       [ 0.73319264,  0.70309419,  0.68684075, ...,  0.67371241,
         0.61473572,  0.64857134],
       [ 0.27081293,  0.05685773, -0.30349841, ...,  0.26092932,
         0.01149813, -0.56784401],
       ...,
       [-0.49397955, -0.4650671 , -0.5849515 , ..., -1.22679824,
        -1.09651385, -0.96128243],
       [-0.56810451, -0.07615307,  0.24284886, ..., -0.25500361,
        -0.54927766, -1.01768834],
       [ 0.67888095,  0.60409168,  0.53403394, ...,  0.56817441,
         0.542762  ,  0.58640596]])
```

```
# Mean for Scaled Data
X_scaled.mean(axis=0)
```

```
array([-3.22973971e-16,  2.15315981e-16, -7.53605932e-16, -
2.15315981e-16,
       -1.72252784e-16, -4.30631961e-17,  3.87568765e-16, -
3.22973971e-17,
        8.61263922e-17, -3.01442373e-16,  7.53605932e-17,
7.75137530e-16,
       -4.52163559e-16, -8.18200726e-16, -3.22973971e-17,
1.93784382e-16,
        6.99776937e-16, -4.09100363e-16, -6.45947942e-17,
4.30631961e-16,
       -1.07657990e-16,  8.18200726e-16, -4.30631961e-17,
8.61263922e-17,
        3.44505569e-16, -2.15315981e-16, -8.61263922e-17, -
3.44505569e-16,
        8.61263922e-17,  4.30631961e-17,  2.36847579e-16,
0.00000000e+00,
       -3.87568765e-16, -3.55271368e-16, -9.68921912e-17,
2.90676574e-16,
       -5.92118946e-16, -3.44505569e-16,  3.22973971e-17,
2.47613378e-16,
       -3.01442373e-16,  1.39955387e-16, -2.15315981e-17,
3.87568765e-16,
       -3.66037167e-16, -2.15315981e-16, -7.64371731e-16, -
1.07657990e-16,
        4.41397760e-16,  2.79910775e-16, -5.05992554e-16,
3.01442373e-16,
       -2.15315981e-17, -1.29189588e-16, -5.38289951e-16,
0.00000000e+00,
        4.30631961e-17,  6.45947942e-17, -6.45947942e-17, -
4.09100363e-16,
        4.30631961e-16, -1.29189588e-16, -2.15315981e-17,
1.07657990e-17,
        3.22973971e-17, -6.56713741e-16,  4.41397760e-16, -
8.61263922e-17,
```

```
        -6.45947942e-17,  8.61263922e-17,  4.73695157e-16,
 2.58379177e-16,
        -3.44505569e-16, -2.47613378e-16, -3.44505569e-16, -
 4.30631961e-17,
        -2.15315981e-16,  2.79910775e-16,  2.79910775e-16, -
 7.53605932e-17,
        -3.12208172e-16, -3.76802966e-16, -3.44505569e-16,
 7.75137530e-16,
        -4.73695157e-16, -2.79910775e-16, -3.55271368e-16, -
 3.44505569e-16,
         1.93784382e-16,  2.36847579e-16, -6.45947942e-17,
 1.29189588e-16,
         5.49055750e-16,  4.30631961e-17,  8.39732324e-16,
 4.30631961e-16,
        -2.90676574e-16,  7.53605932e-16,  8.82795520e-16,
 4.30631961e-17,
        -2.79910775e-16,  1.72252784e-16, -1.29189588e-16, -
 1.07657990e-16,
         3.87568765e-16, -1.72252784e-16,  6.67479540e-16, -
 4.30631961e-16,
        -1.29189588e-16,  1.72252784e-16,  1.72252784e-16,
 3.22973971e-17,
         4.62929358e-16, -1.07657990e-16,  2.15315981e-16,
 0.00000000e+00,
        -4.73695157e-16, -2.58379177e-16,  4.30631961e-16,
 2.36847579e-16,
         1.93784382e-16,  4.30631961e-16,  6.02884745e-16, -
 4.41397760e-16,
        -1.29189588e-16,  6.45947942e-16, -9.25858716e-16,
 2.36847579e-16,
         4.52163559e-16, -4.41397760e-16,  1.50721186e-16,
 1.93784382e-16,
        -3.12208172e-16,  6.45947942e-17,  3.87568765e-16,
 3.22973971e-16,
        -2.15315981e-17, -4.73695157e-16, -1.72252784e-16, -
 3.98334564e-16,
         2.15315981e-17,  2.47613378e-16, -7.42840133e-16,
 4.09100363e-16,
         2.15315981e-17,  5.38289951e-17,  8.61263922e-17, -
 1.50721186e-16,
         2.15315981e-17,  5.16758353e-16,  1.93784382e-16,
 4.19866162e-16,
         7.53605932e-17,  1.39955387e-16,  2.90676574e-16, -
 5.97501846e-16,
         2.69144976e-16,  1.61486985e-17,  8.72029721e-16,
 3.76802966e-16,
         3.44505569e-16,  4.19866162e-16, -1.50721186e-16,
 4.52163559e-16,
         1.83018583e-16, -8.61263922e-17,  0.00000000e+00,
 1.83018583e-16,
```

```
        -2.69144976e-16, -2.58379177e-16, -4.09100363e-16,
3.98334564e-16,
         7.10542736e-16,  8.61263922e-17, -4.95226755e-16, -
1.72252784e-16,
        -3.76802966e-16,  2.36847579e-16,  8.61263922e-17, -
2.15315981e-16,
         8.61263922e-17,  2.15315981e-17,  3.22973971e-16,
4.09100363e-16,
        -2.58379177e-16,  4.25249062e-16,  2.26081780e-16,
2.09933081e-16,
         4.52163559e-16,  3.66037167e-16, -3.33739770e-16, -
2.15315981e-17,
        -1.07657990e-16, -1.07657990e-17, -3.55271368e-16,
1.61486985e-16,
         8.61263922e-17, -2.04550182e-16, -4.52163559e-16,
3.87568765e-16,
        -4.09100363e-16, -2.15315981e-16, -3.33739770e-16, -
4.09100363e-16,
         2.15315981e-17,  2.58379177e-16,  6.35182143e-16, -
4.62929358e-16,
         6.35182143e-16,  3.12208172e-16, -7.53605932e-17, -
2.47613378e-16,
         2.36847579e-16, -6.24416344e-16,  4.52163559e-16, -
5.92118946e-17,
         5.59821549e-16,  3.87568765e-16, -2.15315981e-16, -
3.98334564e-16,
         0.00000000e+00,  3.87568765e-16, -1.29189588e-16, -
5.92118946e-16,
        -2.58379177e-16,  9.68921912e-17,  4.95226755e-16,
7.53605932e-17,
        -5.38289951e-17, -1.72252784e-16,  4.52163559e-16, -
1.72252784e-16,
        -1.29189588e-16,  5.59821549e-16, -3.66037167e-16, -
3.55271368e-16,
        -9.68921912e-17,  7.32074334e-16, -1.00121931e-15, -
1.07657990e-16,
         0.00000000e+00, -3.12208172e-16,  2.58379177e-16,
5.38289951e-17,
        -3.98334564e-16, -7.53605932e-17,  2.36847579e-16, -
2.04550182e-16,
        -1.72252784e-16, -1.61486985e-17,  2.15315981e-17, -
4.19866162e-16,
         1.93784382e-16,  2.26081780e-16,  2.36847579e-16, -
1.72252784e-16,
        -1.39955387e-16,  5.81353147e-16, -4.30631961e-16,
3.87568765e-16,
        -1.72252784e-16, -1.29189588e-16,  1.72252784e-16,
8.61263922e-17,
        -4.30631961e-17, -4.52163559e-16,  9.25858716e-16, -
2.15315981e-17,
```

```
        -5.92118946e-16,  3.98334564e-16, -2.90676574e-16,
3.01442373e-16,
        -2.15315981e-17,  1.72252784e-16, -5.59821549e-16,
5.38289951e-16,
        -2.04550182e-16, -7.53605932e-17,  1.83018583e-16,
3.39122669e-16,
         2.04550182e-16, -1.61486985e-17, -1.61486985e-17,
8.61263922e-17,
        -1.50721186e-16,  1.07657990e-16,  3.44505569e-16, -
4.52163559e-16,
        -2.15315981e-16,  5.38289951e-17, -2.15315981e-17,
2.36847579e-16,
         2.15315981e-17, -2.15315981e-17,  4.84460956e-16, -
6.78245339e-16,
        -2.15315981e-17,  6.67479540e-16,  6.56713741e-16,
5.38289951e-17,
         5.59821549e-16,  1.29189588e-16, -3.33739770e-16,
3.44505569e-16,
         5.38289951e-17, -5.16758353e-16, -5.27524152e-16, -
4.52163559e-16,
         3.06825272e-16, -2.85293674e-16, -5.38289951e-18, -
4.30631961e-16,
         1.18423789e-16,  1.07657990e-16, -1.34572488e-16,
2.15315981e-17,
        -2.15315981e-16,  5.38289951e-17, -1.07657990e-16,
2.47613378e-16,
         1.18423789e-16,  2.15315981e-16,  4.30631961e-17, -
1.29189588e-16,
        -1.93784382e-16,  4.30631961e-17,  2.36847579e-16, -
7.42840133e-16,
        -4.30631961e-17,  1.50721186e-16, -2.15315981e-17,
1.07657990e-16,
        -9.15092917e-17,  4.30631961e-16, -1.39955387e-16, -
1.50721186e-16,
        -1.29189588e-16, -2.69144976e-16,  9.68921912e-17,
2.90676574e-16,
         4.41397760e-16, -5.92118946e-17,  1.02275091e-16,
2.47613378e-16,
         8.07434927e-18,  1.50721186e-16, -1.66869885e-16, -
4.73695157e-16,
        -7.53605932e-17,  1.83018583e-16,  3.01442373e-16,
0.00000000e+00,
        -2.47613378e-16, -2.04550182e-16,  1.07657990e-17,
2.69144976e-16,
         3.44505569e-16, -8.61263922e-17,  4.41397760e-16,
1.07657990e-16,
        -2.15315981e-17, -2.58379177e-16,  2.04550182e-16,
6.24416344e-16,
         6.45947942e-17,  1.50721186e-16,  6.45947942e-17, -
2.96059473e-16,
```

```
         5.92118946e-17,  4.41397760e-16, -3.87568765e-16,
3.76802966e-17,
        -1.95130107e-16,  1.54085499e-16,  1.13713752e-16,
3.64691442e-16,
         3.22973971e-17,  9.15092917e-17, -4.30631961e-17,
1.07657990e-17,
        -5.38289951e-17,  1.29189588e-16, -4.41397760e-16,
1.50721186e-16,
         1.29189588e-16, -1.93784382e-16,  4.30631961e-17, -
2.36847579e-16,
        -3.87568765e-16, -2.04550182e-16, -2.15315981e-17,
4.09100363e-16,
        -1.99167282e-16,  3.49888468e-16,  3.33739770e-16, -
3.39122669e-16,
        -7.64371731e-16, -3.66037167e-16,  2.15315981e-16, -
5.00609655e-16,
         1.07657990e-16,  1.26498139e-16,  2.42230478e-17,
6.14996269e-16,
         4.30631961e-16, -1.61486985e-17,  1.13040890e-16, -
2.36847579e-16,
         2.26081780e-16, -3.76802966e-16,  1.93784382e-16,
4.30631961e-17,
         1.83018583e-16,  1.18423789e-16,  2.47613378e-16,
2.15315981e-17,
        -3.33739770e-16,  3.01442373e-16, -7.53605932e-17,
2.90676574e-16,
         2.15315981e-17,  6.40565042e-16,  5.16758353e-16, -
1.29189588e-16,
        -2.26081780e-16, -2.04550182e-16,  1.66869885e-16,
1.77635684e-16,
         5.92118946e-17, -1.07657990e-16,  2.36847579e-16, -
2.28773229e-16,
         1.20442377e-16, -1.21115239e-16, -2.42230478e-16, -
3.22973971e-17,
         1.18423789e-16,  1.34572488e-16, -7.53605932e-17, -
3.76802966e-16,
         5.38289951e-17, -2.69144976e-16,  1.61486985e-16, -
2.15315981e-16,
        -4.09100363e-16,  8.61263922e-17,  3.22973971e-16,
8.61263922e-17,
        -4.41397760e-16,  1.93784382e-16,  5.59821549e-16,
7.10542736e-16,
        -1.99167282e-16,  7.53605932e-17, -3.33739770e-16, -
1.77635684e-16,
        -3.49888468e-16,  3.44505569e-16,  1.13040890e-16,
6.51330841e-16,
         2.52996277e-16,  4.84460956e-17, -2.77892187e-16,
2.15315981e-16,
         1.99167282e-16,  6.45947942e-17,  9.15092917e-17,
3.20282521e-16,
```

```
         8.61263922e-17, -3.55271368e-16,  8.61263922e-17, -
2.36847579e-16,
         3.12208172e-16,  6.45947942e-17, -1.93784382e-16, -
1.39955387e-16,
         9.68921912e-17,  4.62929358e-16,  3.76802966e-16, -
2.15315981e-17,
         6.02884745e-16, -3.76802966e-17, -2.20698880e-16, -
1.07657990e-16,
         6.45947942e-16, -1.02275091e-16,  3.92951664e-16,
3.92951664e-16,
        -6.99776937e-17, -5.38289951e-18, -1.34572488e-17, -
1.10349440e-16,
        -1.89074345e-16, -8.88178420e-17, -3.60654267e-16, -
2.26081780e-16,
        -4.37360585e-17,  2.52996277e-16, -8.61263922e-17, -
4.84460956e-16,
        -2.04550182e-16,  1.93784382e-16,  6.02884745e-16,
1.72252784e-16,
        -3.44505569e-16,  6.02884745e-16,  1.50721186e-16, -
5.81353147e-16,
        -1.61486985e-16, -6.67479540e-16,  3.44505569e-16,
4.95226755e-16,
         4.19866162e-16, -2.69144976e-17,  5.00609655e-16,
0.00000000e+00,
         2.85293674e-16,  1.93784382e-16, -9.15092917e-17,
1.45338287e-16,
        -2.69144976e-17,  1.02275091e-16,  1.02947953e-16, -
1.48029737e-16,
         2.66453526e-16,  1.18423789e-16, -1.61486985e-16, -
3.12208172e-16,
        -8.61263922e-17,  5.38289951e-17, -1.29189588e-16, -
1.39955387e-16,
        -4.52163559e-16,  1.72252784e-16,  2.58379177e-16, -
9.68921912e-17,
         2.90676574e-16,  4.19866162e-16, -1.93784382e-16, -
4.19866162e-16,
        -3.55271368e-16, -4.30631961e-16, -1.18423789e-16, -
3.55271368e-16,
        -4.84460956e-16,  5.92118946e-17,  5.05992554e-16,
3.01442373e-16,
         3.12208172e-16, -5.11375454e-16, -5.32907052e-16, -
1.64178435e-16,
         1.13040890e-16,  1.60141261e-16,  1.83018583e-16, -
3.05479547e-16,
        -1.29189588e-16, -6.99776937e-17, -3.33739770e-16, -
4.30631961e-17,
        -2.04550182e-16,  4.95226755e-16, -1.61486985e-16, -
3.33739770e-16,
        -1.07657990e-17, -5.70587348e-16,  2.15315981e-17, -
1.18423789e-16,
```

```
         4.52163559e-16, -7.42840133e-16,  1.83018583e-16, -
7.58988831e-16,
        -1.18423789e-16, -2.15315981e-16, -4.57546459e-16, -
1.66869885e-16,
        -3.71420066e-16, -4.84460956e-16, -3.33739770e-16, -
3.22973971e-17,
         4.30631961e-16,  2.09933081e-16,  8.07434927e-18,
2.69144976e-16,
        -3.28356870e-16,  4.84460956e-16, -2.04550182e-16,
4.30631961e-17,
         2.15315981e-17,  5.38289951e-17, -4.19866162e-16,
3.01442373e-16,
        -1.18423789e-16,  5.92118946e-16, -1.72252784e-16, -
1.07657990e-16,
        -5.49055750e-16,  3.66037167e-16, -8.39732324e-16,
1.39955387e-16,
         4.30631961e-17,  1.34572488e-16,  1.11426020e-15,
3.76802966e-16,
         3.60654267e-16, -6.67479540e-16, -2.15315981e-17,
4.09100363e-16,
         4.03717464e-16, -5.92118946e-17, -2.15315981e-17,
1.50721186e-16,
         4.30631961e-17,  1.07657990e-16,  3.39122669e-16, -
1.13040890e-16,
         3.12208172e-16,  7.53605932e-17,  2.15315981e-17,
1.93784382e-16,
         8.61263922e-17, -1.61486985e-16,  3.33739770e-16,
4.95226755e-16,
         1.29189588e-16,  1.07657990e-17, -2.36847579e-16, -
4.30631961e-16,
         1.72252784e-16,  3.22973971e-16,  5.49055750e-16,
1.18423789e-16,
        -2.52996277e-16, -8.55881023e-16, -2.58379177e-16,
1.07657990e-17,
         5.38289951e-16,  4.09100363e-16,  5.92118946e-17, -
3.55271368e-16,
         9.15092917e-17,  4.25249062e-16,  1.29189588e-16, -
6.45947942e-17,
        -2.79910775e-16, -5.38289951e-17,  3.22973971e-17,
4.73695157e-16,
         1.72252784e-16, -1.29189588e-16,  1.83018583e-16,
3.66037167e-16,
         1.18423789e-16, -1.07657990e-16,  5.38289951e-17, -
1.83018583e-16,
         3.01442373e-16,  6.67479540e-16,  1.39955387e-16,
3.66037167e-16,
        -1.50721186e-16,  2.96059473e-16, -2.47613378e-16,
6.40565042e-16,
        -1.61486985e-16,  3.87568765e-16, -3.22973971e-17, -
4.62929358e-16,
```

```
         2.79910775e-16,  3.55271368e-16, -3.22973971e-16, -
4.84460956e-16,
         1.61486985e-17, -7.53605932e-17,  3.44505569e-16, -
5.38289951e-17,
        -7.53605932e-17,  4.52163559e-16,  1.83018583e-16,
2.15315981e-16,
        -1.07657990e-17, -3.12208172e-16, -6.45947942e-17,
6.02884745e-16,
        -1.61486985e-16,  7.53605932e-16, -1.07657990e-17,
0.00000000e+00,
        -2.79910775e-16,  1.39955387e-16, -2.79910775e-16, -
2.15315981e-17,
         7.80520429e-16,  3.12208172e-16,  3.17591071e-16, -
2.20698880e-16,
         1.56104086e-16,  1.00660221e-15,  6.02884745e-16,
1.50721186e-16,
        -4.09100363e-16,  2.04550182e-16,  5.38289951e-17,
5.38289951e-17,
        -1.29189588e-16,  2.69144976e-16,  2.04550182e-16,
2.36847579e-16,
        -1.93784382e-16,  3.98334564e-16,  3.01442373e-16, -
4.41397760e-16,
         3.33739770e-16,  7.42840133e-16,  4.30631961e-17,
2.90676574e-16,
         3.12208172e-16,  1.50721186e-16,  3.66037167e-16, -
3.12208172e-16,
         6.45947942e-17, -1.07657990e-17,  4.84460956e-16,
3.76802966e-16,
        -2.90676574e-16,  2.52996277e-16,  6.35182143e-16,
1.61486985e-17,
         1.93784382e-16,  3.87568765e-16, -3.44505569e-16,
4.30631961e-17,
        -3.44505569e-16, -2.15315981e-17, -1.18423789e-16,
4.95226755e-16,
        -7.53605932e-17,  3.98334564e-16, -4.84460956e-16,
5.59821549e-16,
         3.55271368e-16, -4.09100363e-16, -1.07657990e-16, -
6.45947942e-17,
         2.69144976e-16,  3.01442373e-16,  5.92118946e-16, -
5.38289951e-17,
         7.53605932e-16,  2.15315981e-17, -3.22973971e-17,
3.17591071e-16,
         3.33739770e-16, -9.47390314e-16,  7.53605932e-17, -
4.84460956e-17,
         4.84460956e-17, -3.76802966e-16, -6.45947942e-17,
5.49055750e-16,
         3.22973971e-17, -3.12208172e-16, -3.98334564e-16, -
5.38289951e-16,
         9.68921912e-17,  5.16758353e-16, -2.79910775e-16,
2.26081780e-16,
```

```
         2.58379177e-16, -4.09100363e-16, -1.61486985e-16,
2.26081780e-16,
        -5.70587348e-16, -7.64371731e-16,  2.36847579e-16,
4.30631961e-17,
        -2.47613378e-16, -2.58379177e-16,  2.15315981e-17,
2.58379177e-16,
        -4.73695157e-16, -9.25858716e-16, -3.28356870e-16,
3.82185865e-16,
         3.22973971e-17,  6.99776937e-17,  5.16758353e-16, -
6.99776937e-17,
        -1.72252784e-16,  3.12208172e-16, -2.15315981e-17, -
3.22973971e-16,
         8.61263922e-17,  2.79910775e-16, -3.01442373e-16,
1.07657990e-16,
         6.45947942e-17,  6.02884745e-16, -6.67479540e-16,
3.76802966e-16,
        -3.22973971e-17,  7.32074334e-16, -1.61486985e-16, -
3.22973971e-17,
        -2.15315981e-17,  1.83018583e-16,  5.49055750e-16, -
3.01442373e-16,
         4.30631961e-17,  3.33739770e-16,  4.30631961e-17,
3.66037167e-16,
         2.69144976e-16,  3.55271368e-16, -2.20698880e-16,
7.53605932e-17,
        -6.62096640e-16, -6.24416344e-16, -3.22973971e-17,
1.07657990e-16,
         2.26081780e-16, -2.58379177e-16,  3.22973971e-17,
5.16758353e-16,
         3.55271368e-16, -7.75137530e-16,  7.96669128e-16, -
3.22973971e-17,
        -1.18423789e-16,  3.22973971e-17, -5.70587348e-16,
2.09933081e-16,
         1.18423789e-16, -2.47613378e-16,  4.09100363e-16, -
3.76802966e-16,
         3.76802966e-16,  6.45947942e-17,  1.07657990e-17, -
1.93784382e-16,
        -1.61486985e-16,  5.05992554e-16, -6.45947942e-17,
1.39955387e-16,
        -8.07434927e-17,  3.01442373e-16,  2.85293674e-16, -
3.39122669e-16,
        -6.08267645e-16,  3.98334564e-16,  4.30631961e-17,
2.69144976e-16,
         0.00000000e+00,  3.98334564e-16, -2.36847579e-16, -
1.29189588e-16,
         7.53605932e-17, -1.83018583e-16, -1.72252784e-16,
1.18423789e-16,
        -6.99776937e-17, -1.03351671e-15,  2.15315981e-16,
5.59821549e-16,
        -5.38289951e-17,  3.55271368e-16,  2.47613378e-16, -
6.02884745e-16,
```

```
       -1.83018583e-16, -1.29189588e-16,  3.98334564e-16,
9.68921912e-17,
       -7.21308535e-16,  3.76802966e-16,  5.38289951e-17,
6.89011138e-16,
       -7.05159836e-16, -1.23806689e-16, -1.07657990e-17,
0.00000000e+00,
       -6.67479540e-16,  5.81353147e-16,  1.34572488e-16,
4.95226755e-16,
       -3.22973971e-17,  9.68921912e-17, -2.79910775e-16,
3.33739770e-16,
        2.26081780e-16, -1.50721186e-16,  4.41397760e-16, -
9.25858716e-16,
       -6.67479540e-16, -2.58379177e-16, -5.49055750e-16, -
7.42840133e-16,
       -5.81353147e-16,  3.01442373e-16, -1.07657990e-16,
2.15315981e-16])
```

```
# Standard Deviation for Scaled Data
X_scaled.std(axis=0)
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
```

```
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
```

```
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.])
```

An alternative standardization is scaling features to lie between a given minimum and maximum value, often between zero and one, or so that the maximum absolute value of each feature is scaled to unit size. This can be achieved using MinMaxScaler or MaxAbsScaler, respectively.

```
# from sklearn.preprocessing import MinMaxScaler
# min_max_scaler = MinMaxScaler()
# X_minmax = min_max_scaler.fit_transform(X)
# X_minmax
```

Splitting the Dataset: Train (80%) and Test (20%)

```
# For Scaled Data
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled, y, 
test_size=0.2, random_state=30)

# For Normalised Data
X_train1, X_test1, Y_train1, Y_test1 = train_test_split(X, y, 
test_size=0.2, random_state=30)
```

Checking for Accuracy using Logistic Regression

```
# FOR SCALED DATA
model_1 = LogisticRegression()
model_1.fit(X_train, Y_train)
print("With Scaled Data - ",model_1.score(X_test, Y_test))
```

```
# FOR NORMALISED DATA
model_2 = LogisticRegression()
model_2.fit(X_train1, Y_train1)
print("With Normalised Data - ",model_2.score(X_test1, Y_test1))
```

```
With Scaled Data -  0.6060606060606061
With Normalised Data -  0.7196969696969697

/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/
_logistic.py:763: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

Applying PCA for Dimensionalty Reduction

```
# Information about our independant variable
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 660 entries, 0 to 659
Columns: 900 entries, 0 to 899
dtypes: float64(900)
memory usage: 4.5 MB
```

principal components remove noise by reducing a large number of features to just a couple of principal components.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=600)
```

```
X_norm_pca = pca.fit_transform(X)
X_scal_pca = pca.fit_transform(X_scaled)

pca.n_components_

600

pca.explained_variance_ratio_

array([4.81292311e-01, 1.73876913e-01, 4.64942877e-02, 3.72512043e-02,
       3.10873974e-02, 2.58693769e-02, 2.03099825e-02, 1.78104322e-02,
       1.64642226e-02, 1.50975227e-02, 9.44433291e-03, 7.88740566e-03,
       7.42254416e-03, 6.84056652e-03, 6.17258777e-03, 5.86154157e-03,
       5.32298355e-03, 3.80276746e-03, 3.44706261e-03, 3.25753128e-03,
       3.24404098e-03, 2.91270638e-03, 2.76310198e-03, 2.72490380e-03,
       2.50742841e-03, 2.18141054e-03, 2.02618805e-03, 1.90603963e-03,
       1.85461846e-03, 1.62028433e-03, 1.53798910e-03, 1.49637807e-03,
       1.45913234e-03, 1.39609104e-03, 1.30956532e-03, 1.27694536e-03,
       1.25589073e-03, 1.21769645e-03, 1.11035693e-03, 1.07504999e-03,
       9.88536979e-04, 9.28592213e-04, 8.65907565e-04, 8.38752861e-04,
       8.17808816e-04, 7.68910267e-04, 7.43785972e-04, 7.30805875e-04,
       7.15396848e-04, 6.71780563e-04, 6.65126997e-04, 6.41459082e-04,
       6.18237205e-04, 5.91175063e-04, 5.90781720e-04, 5.66563306e-04,
       5.40680995e-04, 5.31743783e-04, 5.07123216e-04, 4.97214773e-04,
       4.81368043e-04, 4.61432332e-04, 4.49777086e-04, 4.41159519e-04,
       4.28501522e-04, 4.16178013e-04, 4.05914437e-04, 3.96219599e-04,
       3.85090247e-04, 3.70907893e-04, 3.61236108e-04, 3.54280281e-04,
       3.40169598e-04, 3.34204377e-04, 3.21648957e-04, 3.16693300e-04,
       3.13267123e-04, 3.02562282e-04, 2.98475602e-04, 2.86960878e-04,
       2.84693179e-04, 2.76169863e-04, 2.73570562e-04, 2.61272164e-04,
       2.58439203e-04, 2.54519608e-04, 2.48976795e-04, 2.41938919e-04,
       2.39111596e-04, 2.33210920e-04, 2.26967024e-04, 2.24704467e-04,
       2.21573576e-04, 2.15472823e-04, 2.08986898e-04, 2.06082590e-04,
       2.03009643e-04, 1.99019566e-04, 1.93914270e-04, 1.88693972e-04,
       1.85505752e-04, 1.82119164e-04, 1.75781614e-04, 1.72029239e-04,
       1.65979190e-04, 1.64048465e-04, 1.62373696e-04, 1.55324240e-04,
       1.53898111e-04, 1.51160539e-04, 1.48699349e-04, 1.48150536e-04,
       1.46586523e-04, 1.41595939e-04, 1.39278302e-04, 1.37516554e-04,
       1.33750615e-04, 1.28431316e-04, 1.27770308e-04, 1.25218304e-04,
       1.23193239e-04, 1.20790329e-04, 1.19425847e-04, 1.17737544e-04,
       1.15572174e-04, 1.14100346e-04, 1.13098938e-04, 1.11161911e-04,
       1.09725462e-04, 1.07416796e-04, 1.05188264e-04, 1.03620805e-04,
       1.00909551e-04, 1.00834876e-04, 9.90571261e-05, 9.79508604e-05,
       9.52282422e-05, 9.39323368e-05, 9.33399938e-05, 9.17181177e-05,
       9.05376878e-05, 8.90522782e-05, 8.83849352e-05, 8.71857967e-05,
       8.53903911e-05, 8.23996073e-05, 8.18410720e-05, 8.11671219e-05,
       7.92338591e-05, 7.85404695e-05, 7.71000086e-05, 7.62425253e-05,
       7.53508836e-05, 7.44714070e-05, 7.32643112e-05, 7.22691017e-05,
       7.18897790e-05, 7.04941841e-05, 7.00647753e-05, 6.82832406e-05,
       6.75251979e-05, 6.69309772e-05, 6.56363280e-05, 6.39746632e-05,
       6.35701288e-05, 6.26478574e-05, 6.13760603e-05, 6.00391490e-05,
```

```
5.98336405e-05, 5.82041079e-05, 5.77211175e-05, 5.73200644e-05,
5.66033637e-05, 5.56344901e-05, 5.51178208e-05, 5.40363098e-05,
5.29165653e-05, 5.23807893e-05, 5.15351568e-05, 5.11976048e-05,
5.01724200e-05, 4.97865994e-05, 4.85093495e-05, 4.82731922e-05,
4.76689006e-05, 4.73102849e-05, 4.68421253e-05, 4.61817811e-05,
4.51227627e-05, 4.43619997e-05, 4.40349262e-05, 4.37009208e-05,
4.32384110e-05, 4.25016878e-05, 4.19039361e-05, 4.14675035e-05,
4.10375126e-05, 4.04543290e-05, 4.02044149e-05, 3.91319619e-05,
3.82075039e-05, 3.78569366e-05, 3.73187163e-05, 3.70195585e-05,
3.67394666e-05, 3.58747014e-05, 3.56438567e-05, 3.52893727e-05,
3.50148200e-05, 3.42875766e-05, 3.38154312e-05, 3.33341385e-05,
3.26809380e-05, 3.25627120e-05, 3.20622091e-05, 3.15582708e-05,
3.12504646e-05, 3.06774580e-05, 3.05365150e-05, 2.98688182e-05,
2.95791943e-05, 2.89265573e-05, 2.88630983e-05, 2.86420081e-05,
2.83433787e-05, 2.79616807e-05, 2.77728086e-05, 2.73382319e-05,
2.68856149e-05, 2.67660197e-05, 2.62960215e-05, 2.60908302e-05,
2.60491191e-05, 2.58626212e-05, 2.53090658e-05, 2.49599422e-05,
2.48053068e-05, 2.45469706e-05, 2.40100338e-05, 2.39128956e-05,
2.38062984e-05, 2.32037698e-05, 2.27600272e-05, 2.24445703e-05,
2.21564857e-05, 2.19171269e-05, 2.16814919e-05, 2.14702202e-05,
2.13289815e-05, 2.12323041e-05, 2.09096160e-05, 2.08601009e-05,
2.05010266e-05, 2.02047772e-05, 1.99947282e-05, 1.97129316e-05,
1.95480960e-05, 1.94503385e-05, 1.91618232e-05, 1.88152024e-05,
1.84850261e-05, 1.83900552e-05, 1.81926894e-05, 1.78841159e-05,
1.76563563e-05, 1.76105901e-05, 1.74484398e-05, 1.72904677e-05,
1.69233100e-05, 1.67557904e-05, 1.65164390e-05, 1.63944240e-05,
1.62996057e-05, 1.60282515e-05, 1.58957025e-05, 1.54522383e-05,
1.54428801e-05, 1.52958712e-05, 1.52297541e-05, 1.50658779e-05,
1.48822647e-05, 1.46994276e-05, 1.45077073e-05, 1.42727030e-05,
1.40947191e-05, 1.39676842e-05, 1.39604223e-05, 1.37082771e-05,
1.34547548e-05, 1.33039633e-05, 1.32626278e-05, 1.29834859e-05,
1.28642176e-05, 1.28422548e-05, 1.26934938e-05, 1.25927951e-05,
1.23753491e-05, 1.22987278e-05, 1.20910651e-05, 1.20278820e-05,
1.19136163e-05, 1.18679693e-05, 1.17749705e-05, 1.16769532e-05,
1.14321988e-05, 1.12190397e-05, 1.11793047e-05, 1.10990011e-05,
1.09340411e-05, 1.08087155e-05, 1.06282124e-05, 1.05504832e-05,
1.04617204e-05, 1.04376842e-05, 1.02334297e-05, 1.01684972e-05,
1.01059129e-05, 9.97302085e-06, 9.84071365e-06, 9.78394741e-06,
9.62167292e-06, 9.58070101e-06, 9.56260686e-06, 9.46547914e-06,
9.34719052e-06, 9.21700516e-06, 9.14625688e-06, 9.12351603e-06,
9.00694737e-06, 8.81662840e-06, 8.73774896e-06, 8.70743863e-06,
8.59358965e-06, 8.50174067e-06, 8.40328047e-06, 8.33346962e-06,
8.28692977e-06, 8.16680658e-06, 8.07813584e-06, 8.05269405e-06,
7.97226073e-06, 7.87552845e-06, 7.81095578e-06, 7.71338845e-06,
7.64354433e-06, 7.53289828e-06, 7.39122034e-06, 7.35246360e-06,
7.31848560e-06, 7.25888525e-06, 7.20352178e-06, 7.05676508e-06,
6.97232451e-06, 6.93111500e-06, 6.82531565e-06, 6.75175865e-06,
6.69361737e-06, 6.62202571e-06, 6.60456331e-06, 6.58526382e-06,
6.48081837e-06, 6.40899611e-06, 6.37224811e-06, 6.30945046e-06,
6.29000866e-06, 6.21457725e-06, 6.15384622e-06, 6.07494010e-06,
```

```
5.97299177e-06, 5.92762071e-06, 5.82567965e-06, 5.77304898e-06,
5.65790180e-06, 5.63553699e-06, 5.57920789e-06, 5.52269127e-06,
5.44949180e-06, 5.41089482e-06, 5.40405483e-06, 5.32241219e-06,
5.27854587e-06, 5.23376698e-06, 5.16015076e-06, 5.10949494e-06,
5.06519474e-06, 5.00479672e-06, 4.98389464e-06, 4.93083927e-06,
4.87068995e-06, 4.82592757e-06, 4.77350792e-06, 4.71401379e-06,
4.61893601e-06, 4.58821926e-06, 4.52580218e-06, 4.50368695e-06,
4.44250942e-06, 4.37002381e-06, 4.33953201e-06, 4.28643190e-06,
4.24304993e-06, 4.24151340e-06, 4.18092559e-06, 4.12927978e-06,
4.07720680e-06, 4.06003000e-06, 4.02015109e-06, 3.96789532e-06,
3.92350685e-06, 3.86363315e-06, 3.84649797e-06, 3.80067365e-06,
3.78660724e-06, 3.76883383e-06, 3.71127379e-06, 3.68017736e-06,
3.61768351e-06, 3.60813560e-06, 3.57331646e-06, 3.52458007e-06,
3.51867631e-06, 3.45244634e-06, 3.41822022e-06, 3.39673406e-06,
3.31654727e-06, 3.29883309e-06, 3.27882337e-06, 3.24280404e-06,
3.19206913e-06, 3.17023359e-06, 3.16609071e-06, 3.11558181e-06,
3.08269110e-06, 3.06933276e-06, 3.02438593e-06, 2.97070906e-06,
2.92867545e-06, 2.90149851e-06, 2.84972657e-06, 2.82924511e-06,
2.82403789e-06, 2.79470430e-06, 2.76623497e-06, 2.74210232e-06,
2.70967858e-06, 2.68272879e-06, 2.65089757e-06, 2.62492520e-06,
2.59375947e-06, 2.55883489e-06, 2.50771506e-06, 2.49248988e-06,
2.46236519e-06, 2.45233715e-06, 2.40009812e-06, 2.37282798e-06,
2.32069568e-06, 2.30244252e-06, 2.28547994e-06, 2.27847644e-06,
2.25403097e-06, 2.20133592e-06, 2.19475607e-06, 2.18671025e-06,
2.15423911e-06, 2.13077572e-06, 2.12178695e-06, 2.09384305e-06,
2.08307724e-06, 2.04573028e-06, 2.02916874e-06, 2.00069643e-06,
1.98963867e-06, 1.95499842e-06, 1.94545008e-06, 1.91564033e-06,
1.88145696e-06, 1.86810798e-06, 1.84326626e-06, 1.82321295e-06,
1.80014028e-06, 1.78366184e-06, 1.75926753e-06, 1.73492884e-06,
1.70080078e-06, 1.68617235e-06, 1.67348201e-06, 1.65745335e-06,
1.63494157e-06, 1.61164510e-06, 1.60958287e-06, 1.58659463e-06,
1.56474349e-06, 1.53602525e-06, 1.52290967e-06, 1.50939233e-06,
1.50410168e-06, 1.46942443e-06, 1.46585776e-06, 1.45726130e-06,
1.43203700e-06, 1.42037310e-06, 1.41712893e-06, 1.37846657e-06,
1.36424006e-06, 1.34653232e-06, 1.33306539e-06, 1.32706261e-06,
1.32408871e-06, 1.29438507e-06, 1.27982650e-06, 1.26215813e-06,
1.23887017e-06, 1.23338311e-06, 1.21364646e-06, 1.20634178e-06,
1.19217806e-06, 1.17430703e-06, 1.16152904e-06, 1.14785321e-06,
1.13518283e-06, 1.11822364e-06, 1.09863229e-06, 1.09182858e-06,
1.07700835e-06, 1.06675874e-06, 1.05566586e-06, 1.04096291e-06,
1.03439837e-06, 1.02201382e-06, 9.99772305e-07, 9.91964854e-07,
9.59117382e-07, 9.53327706e-07, 9.39191392e-07, 9.24804862e-07,
9.13508696e-07, 9.06172696e-07, 9.00832131e-07, 8.85261653e-07,
8.74263409e-07, 8.57301699e-07, 8.49722059e-07, 8.45149946e-07,
8.36092323e-07, 8.17401530e-07, 8.08052196e-07, 8.00950265e-07,
7.96105576e-07, 7.84700088e-07, 7.76283408e-07, 7.58007211e-07,
7.49369101e-07, 7.42638527e-07, 7.25185439e-07, 7.18882294e-07,
7.12289219e-07, 7.06516221e-07, 7.02744112e-07, 6.87514984e-07,
6.65018388e-07, 6.58435638e-07, 6.50905388e-07, 6.43156069e-07,
6.36301188e-07, 6.30139209e-07, 6.19936354e-07, 6.10366090e-07,
```

```
       6.02046402e-07, 5.77352237e-07, 5.73026335e-07, 5.66576113e-07,
       5.58568229e-07, 5.56518461e-07, 5.46823570e-07, 5.38134675e-07,
       5.31505022e-07, 5.15332797e-07, 5.13196311e-07, 5.11859892e-07,
       5.04283887e-07, 4.86242509e-07, 4.82786152e-07, 4.69118090e-07,
       4.63874488e-07, 4.58944009e-07, 4.56651319e-07, 4.50164718e-07,
       4.46274351e-07, 4.37353583e-07, 4.27098663e-07, 4.23757306e-07,
       4.13418435e-07, 4.03682640e-07, 4.01081213e-07, 3.93488274e-07,
       3.88093637e-07, 3.84725861e-07, 3.76043299e-07, 3.69697664e-
07])
```

```
# Information about our independant variable after PCA
X_scal_pca.shape
```

```
(660, 600)
```

PCA created 659 columns out of 40000 original columns

```
X_norm_pca
```

```
array([[-4.95746244e-01,  1.30447824e+00, -2.87213365e-01, ...,
         1.36151741e-03, -5.76273096e-03,  1.36905889e-03],
       [-1.49484419e+00,  1.63354795e+00, -5.68459603e-01, ...,
        -2.26894778e-03, -7.80388632e-04, -3.01340477e-03],
       [-4.37254157e+00, -2.43472044e+00,  6.62649739e-01, ...,
        -1.95871876e-03,  2.04288940e-03,  1.73934590e-03],
       ...,
       [ 1.09151427e+00, -2.61856980e+00, -2.00335639e+00, ...,
        -6.11697474e-04,  2.01237781e-03,  1.34691755e-03],
       [ 4.08838319e+00,  5.65597923e+00, -1.10607646e+00, ...,
         3.89326892e-04,  9.48129158e-04,  5.73100460e-04],
       [-1.32611832e+00, -1.14561835e+00, -4.29202793e-01, ...,
         1.12644329e-03,  3.88965851e-03, -1.96893802e-03]])
```

Again Training and checking accuracy of our model after PCA

```
# FOR SCALED DATA
X_train, X_test, Y_train, Y_test = train_test_split(X_scal_pca, y,
test_size=0.2, random_state=30)
model_1 = LogisticRegression(max_iter=5000)
model_1.fit(X_train, Y_train)
print("With Scaled Data - ",model_1.score(X_test, Y_test))

# FOR NORMALISED DATA
X_train1, X_test1, Y_train1, Y_test1 = train_test_split(X_norm_pca, y,
test_size=0.2, random_state=30)
model_2 = LogisticRegression()
model_2.fit(X_train1, Y_train1)
print("\nWith Normalised Data - ",model_2.score(X_test1, Y_test1))
```

```
With Scaled Data -  0.6363636363636364

With Normalised Data -  0.7121212121212122
```

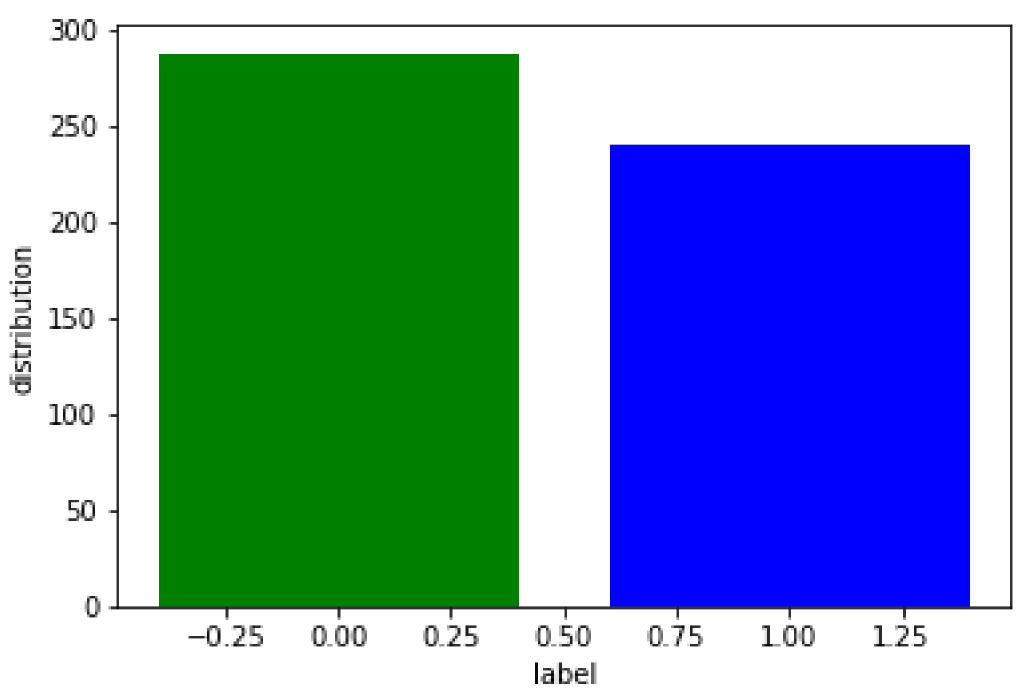
we can do the same by converting our image data to CSV file then apply the PCA

```
# test30=x_test
# test30['label'] = y_test
# test30.to_csv('test25.csv')

#Heat Map
# dataset25 = pd.read_csv('train25.csv')
# sns.heatmap(dataset25.corr(),annot=True)
```

Applying Classification

```
one = 0
zero = 0

for i in Y_train1:
    if i == 1:
        one+=1
    else:
        zero+=1

x = [1,0]
y = [one,zero]

plt.bar(x,y,color=['blue', 'g'])
plt.xlabel('label')
plt.ylabel('distribution')
```

```
Text(0, 0.5, 'distribution')
```



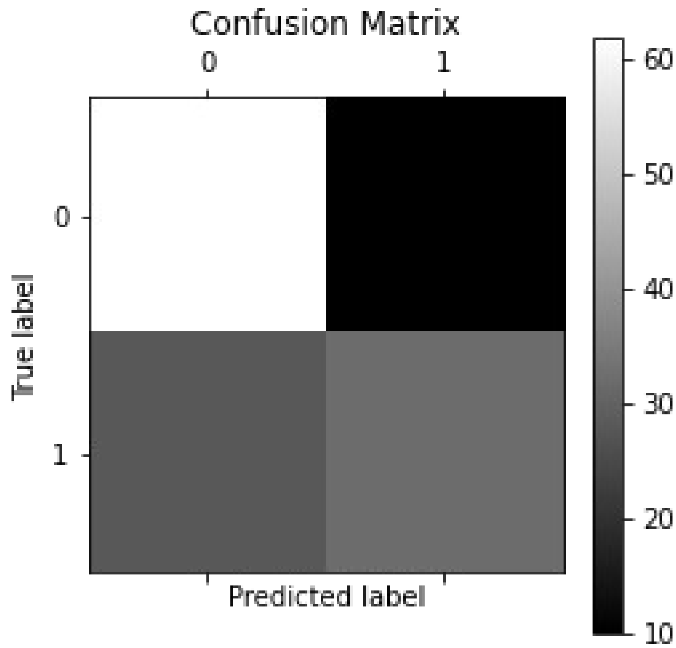
Algorithms Applied

1) Logistic Regression

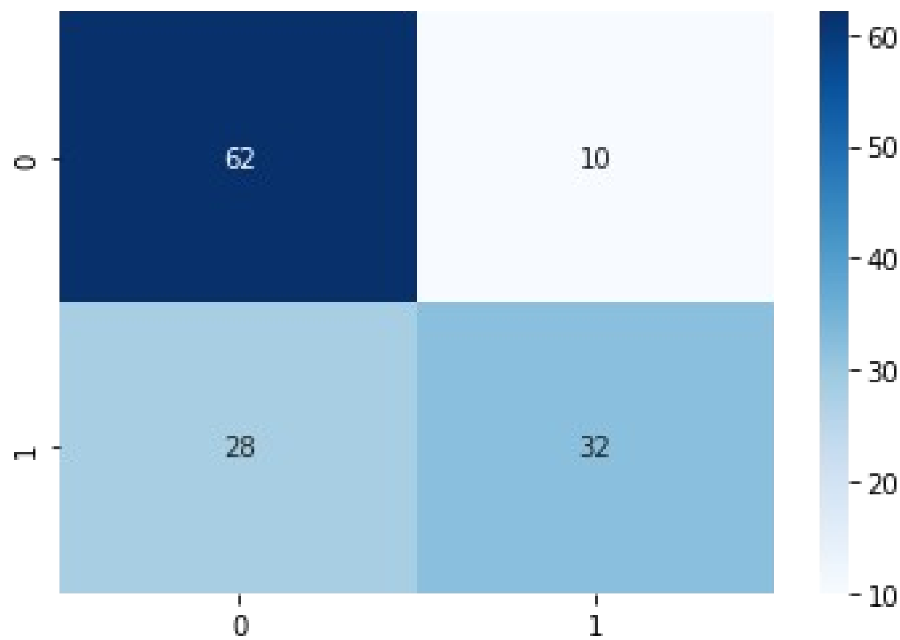
```
from sklearn.linear_model import LogisticRegression
print("training Logistic Regression Model")
model_1 = LogisticRegression()
model_1.fit(X_train1, Y_train1)
print("model trained")

# MODEL EVALUATION
target_names = ['Benign', 'Malignant']
# ACCURACY SCORE
Y_pred_1 = model_1.predict(X_test1)
logisticReg = accuracy_score(Y_pred_1, Y_test1)
print('Accuracy: ', round(logisticReg * 100, 2))
linear = r2_score(Y_test1,Y_pred_1)
print('R2 Score: ', linear)
model_eval = classification_report(Y_test1,
Y_pred_1,target_names=target_names)
print('\nModel Evaluation Table: \n', model_eval)
cm = confusion_matrix(Y_test1, Y_pred_1)
print(cm)

# PLOT CONFUSION MATRIX
plt.matshow(cm)
plt.title('Confusion Matrix')
plt.colorbar()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()

# PLOT HEATMAP
sns.heatmap(cm,cmap='Blues',annot=True,fmt='g')
cm_flat = cm.flatten()
TN = cm_flat[0]
FP = cm_flat[1]
FN = cm_flat[2]
TP = cm_flat[3]
recall = TP/(TP+FN)
precision =TP/(TP+FP)
specificity = TN/(TN+FP)
print('Specificity = ',specificity)
```

```
training Logistic Regression Model
model trained
Accuracy:  71.21
R2 Score:  -0.16111111111111143

Model Evaluation Table: 
               precision    recall  f1-score   support
```

```
       Benign       0.69      0.86      0.77        72
    Malignant       0.76      0.53      0.63        60

     accuracy                           0.71       132
    macro avg       0.73      0.70      0.70       132
 weighted avg       0.72      0.71      0.70       132

[[62 10]
 [28 32]]
```




```
Specificity =  0.8611111111111112
```

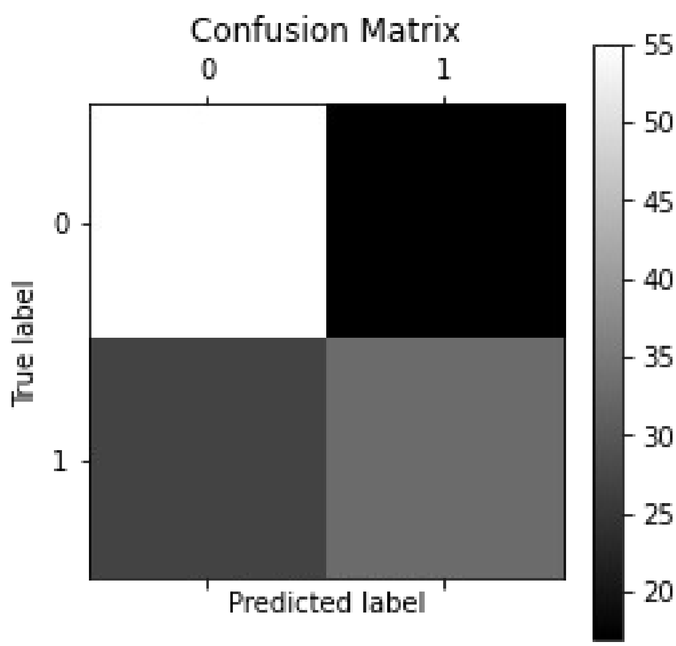


2) Decision Tree

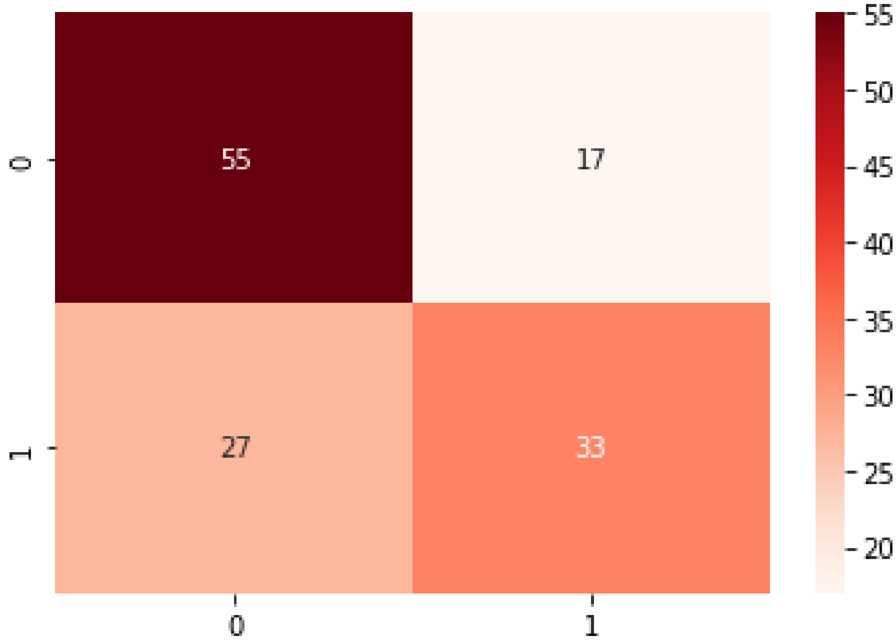
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
print("training Decision Tree Classifier Model")
model.fit(X_train1, Y_train1)
print("model trained")

# MODEL EVALUATION
target_names = ['Benign', 'Malignant']
# ACCURACY SCORE
xtest_pred = model.predict(X_test1)
DecisionT = accuracy_score(xtest_pred, Y_test1)
print('Accuracy: ', round(DecisionT * 100, 2))
linear = r2_score(Y_test1,xtest_pred)
print('R2 Score: ', linear)
model_eval = classification_report(Y_test1,
xtest_pred ,target_names=target_names)
print('\nModel Evaluation Table: \n', model_eval)
cm = confusion_matrix(Y_test1, xtest_pred)
print(cm)

# PLOT CONFUSION MATRIX
plt.matshow(cm)
plt.title('Confusion Matrix')
plt.colorbar()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```

```python
# PLOT HEATMAP
sns.heatmap(cm,cmap='Reds',annot=True,fmt='g')
cm_flat = cm.flatten()
TN = cm_flat[0]
FP = cm_flat[1]
FN = cm_flat[2]
TP = cm_flat[3]
recall = TP/(TP+FN)
precision =TP/(TP+FP)
specificity = TN/(TN+FP)
print('Specificity = ',specificity)
```

```
training Decision Tree Classifier Model
model trained
Accuracy:  66.67
R2 Score:  -0.3444444444444448

Model Evaluation Table:
               precision    recall  f1-score   support

      Benign       0.67      0.76      0.71        72
   Malignant       0.66      0.55      0.60        60

    accuracy                           0.67       132
   macro avg       0.67      0.66      0.66       132
weighted avg       0.67      0.67      0.66       132

[[55 17]
 [27 33]]
```



```
Specificity =  0.7638888888888888
```

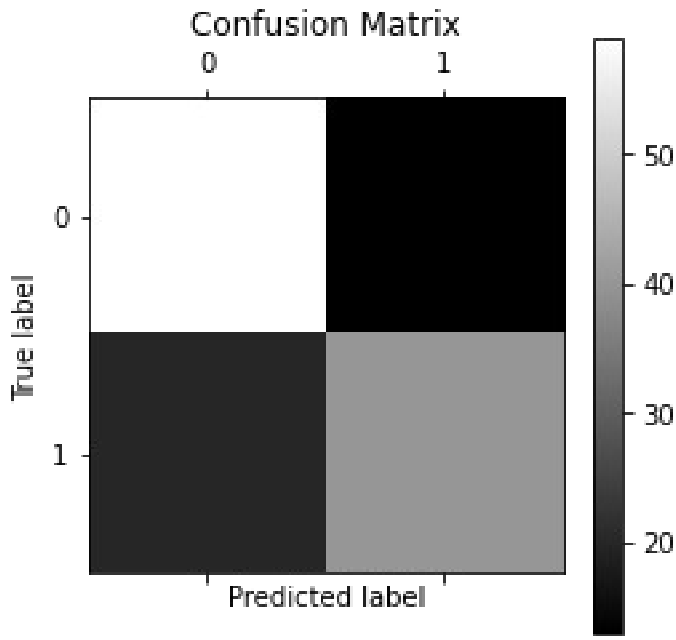


3) Gaussian Naive Bayes Classifier

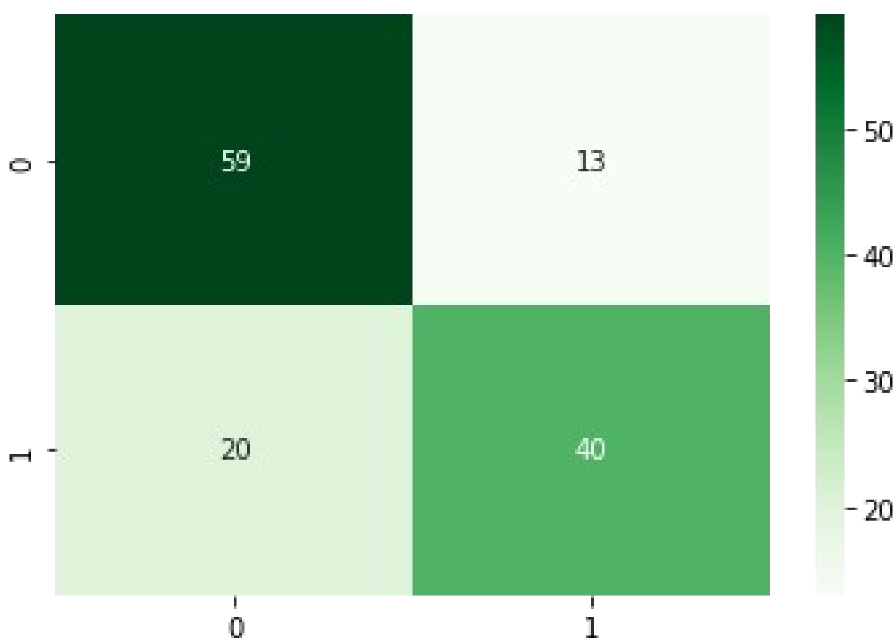
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
print("training Gaussian Model")
model.fit(X_train1, Y_train1)
print("model trained")

# MODEL EVALUATION
target_names = ['Benign', 'Malignant']
# ACCURACY SCORE
xtest_pred = model.predict(X_test1)
GaussianNB = accuracy_score(xtest_pred, Y_test1)
print('Accuracy: ', round(GaussianNB * 100, 2))
linear = r2_score(Y_test1,xtest_pred)
print('R2 Score: ', linear)
model_eval = classification_report(Y_test1,
xtest_pred ,target_names=target_names)
print('\nModel Evaluation Table: \n', model_eval)
cm = confusion_matrix(Y_test1, xtest_pred)
print(cm)

# PLOT CONFUSION MATRIX
plt.matshow(cm)
plt.title('Confusion Matrix')
plt.colorbar()
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
plt.show()

# PLOT HEATMAP
sns.heatmap(cm,cmap='Greens',annot=True,fmt='g')
cm_flat = cm.flatten()
TN = cm_flat[0]
FP = cm_flat[1]
FN = cm_flat[2]
TP = cm_flat[3]
recall = TP/(TP+FN)
precision =TP/(TP+FP)
specificity = TN/(TN+FP)
print('Specificity = ',specificity)
```

```
training Gaussian Model
model trained
Accuracy:  75.0
R2 Score:  -0.008333333333333526

Model Evaluation Table:
               precision    recall  f1-score   support

      Benign       0.75      0.82      0.78        72
   Malignant       0.75      0.67      0.71        60

    accuracy                           0.75       132
   macro avg       0.75      0.74      0.74       132
weighted avg       0.75      0.75      0.75       132

[[59 13]
 [20 40]]
```




```
Specificity =  0.8194444444444444
```

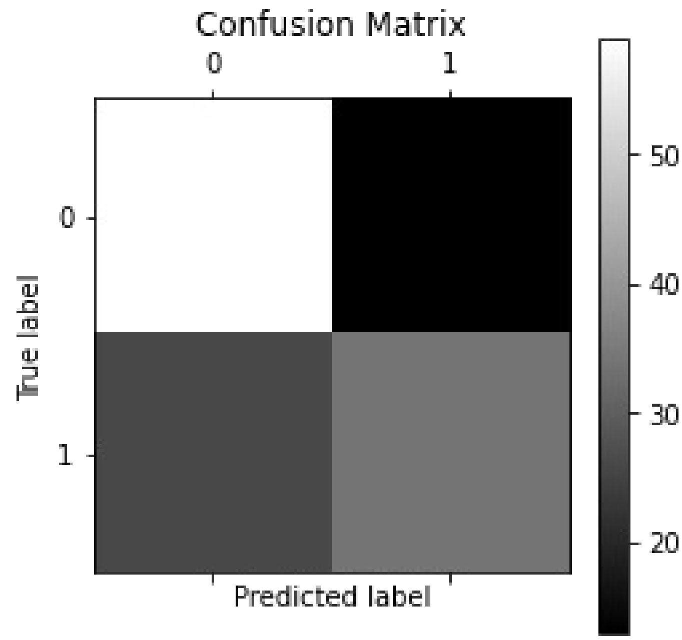


4) Random Forest Classifier

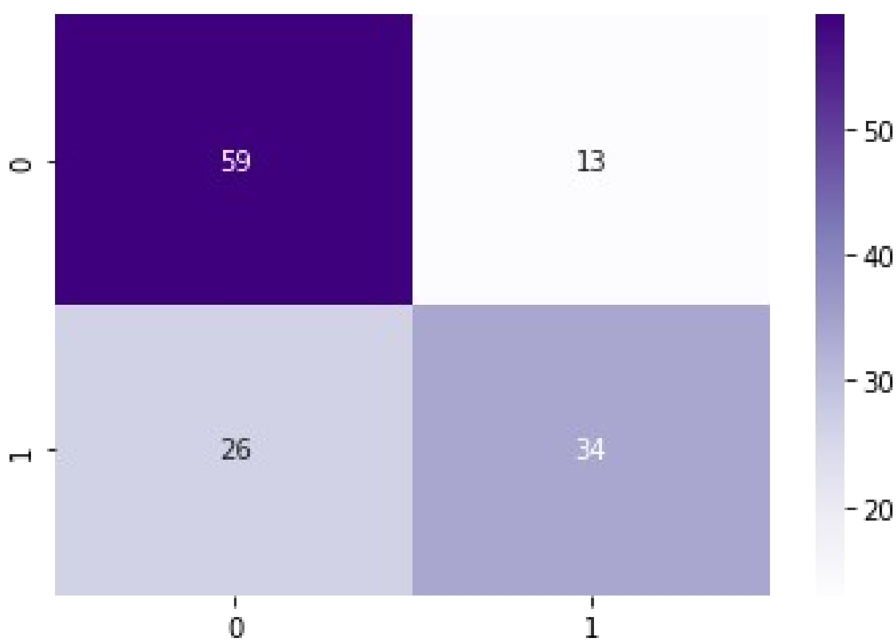
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(max_depth=100, random_state=0)
print("training random forest model")
model.fit(X_train1, Y_train1)
print("model trained")

# MODEL EVALUATION
```

```python
target_names = ['Benign', 'Malignant']
# ACCURACY SCORE
xtest_pred = model.predict(X_test1)
RandomF = accuracy_score(xtest_pred, Y_test1)
print('Accuracy: ', round(RandomF * 100, 2))
linear = r2_score(Y_test1,xtest_pred)
print('R2 Score: ', linear)
model_eval = classification_report(Y_test1,
xtest_pred ,target_names=target_names)
print('\nModel Evaluation Table: \n', model_eval)
cm = confusion_matrix(Y_test1, xtest_pred)
print(cm)

# PLOT CONFUSION MATRIX
plt.matshow(cm)
plt.title('Confusion Matrix')
plt.colorbar()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()

# PLOT HEATMAP
sns.heatmap(cm,cmap='Purples',annot=True,fmt='g')
cm_flat = cm.flatten()
TN = cm_flat[0]
FP = cm_flat[1]
FN = cm_flat[2]
TP = cm_flat[3]
recall = TP/(TP+FN)
precision =TP/(TP+FP)
specificity = TN/(TN+FP)
print('Specificity = ',specificity)
```

```
training random forest model
model trained
Accuracy:  70.45
R2 Score:  -0.1916666666666687

Model Evaluation Table:
               precision    recall  f1-score   support

      Benign       0.69      0.82      0.75        72
   Malignant       0.72      0.57      0.64        60

    accuracy                           0.70       132
   macro avg       0.71      0.69      0.69       132
weighted avg       0.71      0.70      0.70       132

[[59 13]
 [26 34]]
```




```
Specificity =  0.8194444444444444
```



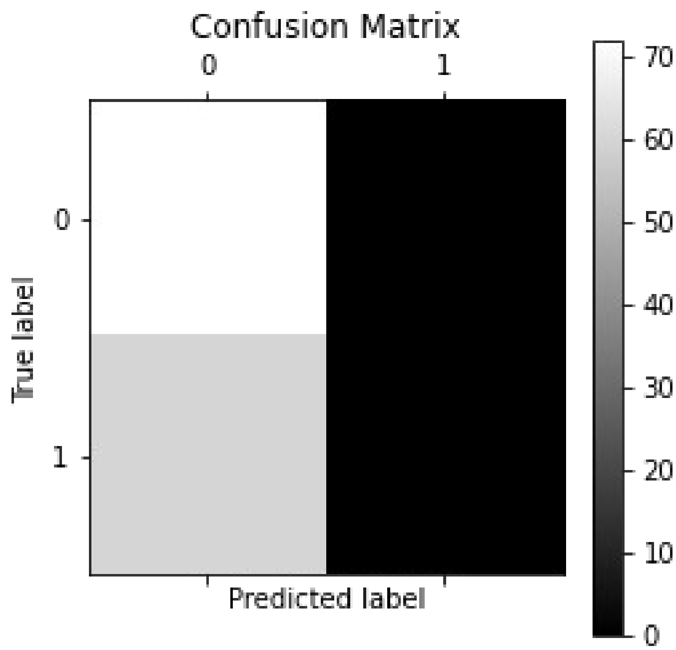
5) K Nearest Neighbor Classifier

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=X_train1.shape[0])
print("training knn model")
model.fit(X_train1, Y_train1)
print("model trained")

# MODEL EVALUATION
```

```
target_names = ['Benign', 'Malignant']
# ACCURACY SCORE
xtest_pred = model.predict(X_test1)
KNN = accuracy_score(xtest_pred, Y_test1)
print('Accuracy: ', round(KNN * 100, 2))
linear = r2_score(Y_test1,xtest_pred)
print('R2 Score: ', linear)
model_eval = classification_report(Y_test1,
xtest_pred ,target_names=target_names)
print('\nModel Evaluation Table: \n', model_eval)
cm = confusion_matrix(Y_test1, xtest_pred)
print(cm)

# PLOT CONFUSION MATRIX
plt.matshow(cm)
plt.title('Confusion Matrix')
plt.colorbar()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()

# PLOT HEATMAP
sns.heatmap(cm,cmap='Oranges',annot=True,fmt='g')
cm_flat = cm.flatten()
TN = cm_flat[0]
FP = cm_flat[1]
FN = cm_flat[2]
TP = cm_flat[3]
recall = TP/(TP+FN)
precision =TP/(TP+FP)
specificity = TN/(TN+FP)
print('Specificity = ',specificity)
```

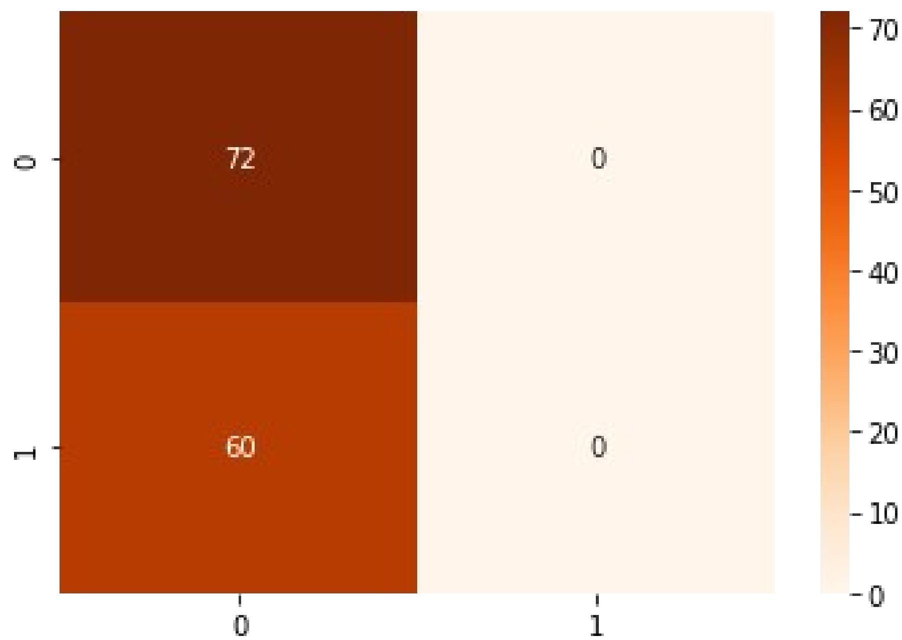
```
training knn model
model trained
Accuracy:  54.55
R2 Score:  -0.8333333333333337

Model Evaluation Table:
               precision    recall  f1-score   support

      Benign       0.55      1.00      0.71        72
   Malignant       0.00      0.00      0.00        60

    accuracy                           0.55       132
   macro avg       0.27      0.50      0.35       132
weighted avg       0.30      0.55      0.39       132

[[72  0]
 [60  0]]
```

```
/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/
_classification.py:1248: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_classifica
tion.py:1248: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_classifica
tion.py:1248: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```




```
Specificity =  1.0
```

```
/var/folders/kv/9cymplv97pb6645dxkl_mgp80000gn/T/
ipykernel_94570/3361439967.py:36: RuntimeWarning: invalid value
encountered in long_scalars
  precision =TP/(TP+FP)
```

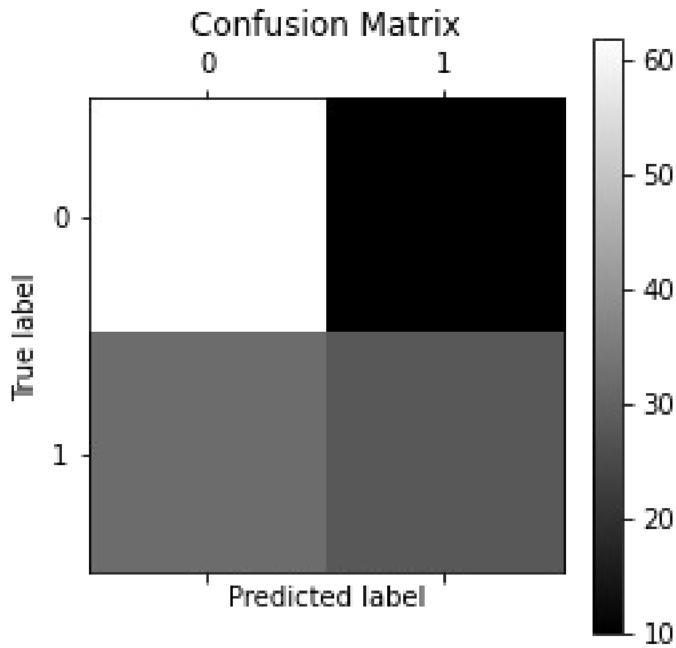


6) Support Vector Machine

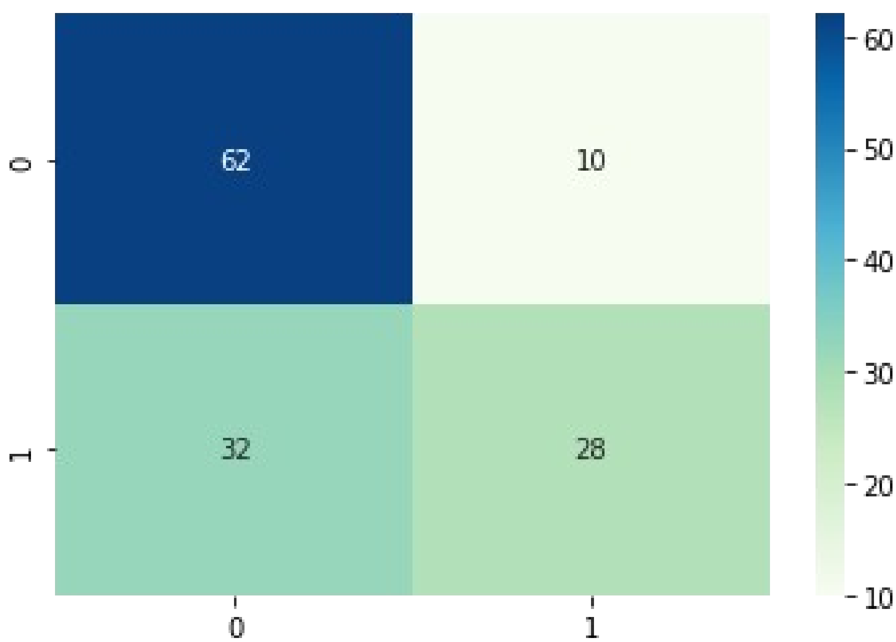
```
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
print("training svm model")
model = clf.fit(X_train1, Y_train1)
print("model trained")

# MODEL EVALUATION
target_names = ['Benign', 'Malignant']
# ACCURACY SCORE
xtest_pred = model.predict(X_test1)
SVM = accuracy_score(xtest_pred, Y_test1)
print('Accuracy: ', round(SVM * 100, 2))
linear = r2_score(Y_test1,xtest_pred)
print('R2 Score: ', linear)
model_eval = classification_report(Y_test1,
xtest_pred ,target_names=target_names)
print('\nModel Evaluation Table: \n', model_eval)
cm = confusion_matrix(Y_test1, xtest_pred)
print(cm)

# PLOT CONFUSION MATRIX
plt.matshow(cm)
plt.title('Confusion Matrix')
plt.colorbar()
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```python
plt.show()

# PLOT HEATMAP
sns.heatmap(cm,cmap='GnBu',annot=True,fmt='g')
cm_flat = cm.flatten()
TN = cm_flat[0]
FP = cm_flat[1]
FN = cm_flat[2]
TP = cm_flat[3]
recall = TP/(TP+FN)
precision =TP/(TP+FP)
specificity = TN/(TN+FP)
print('Specificity = ',specificity)
```

```
training svm model
model trained
Accuracy:  68.18
R2 Score:  -0.28333333333333366

Model Evaluation Table:
               precision    recall  f1-score   support

      Benign       0.66      0.86      0.75        72
   Malignant       0.74      0.47      0.57        60

    accuracy                           0.68       132
   macro avg       0.70      0.66      0.66       132
weighted avg       0.69      0.68      0.67       132

[[62 10]
 [32 28]]
```



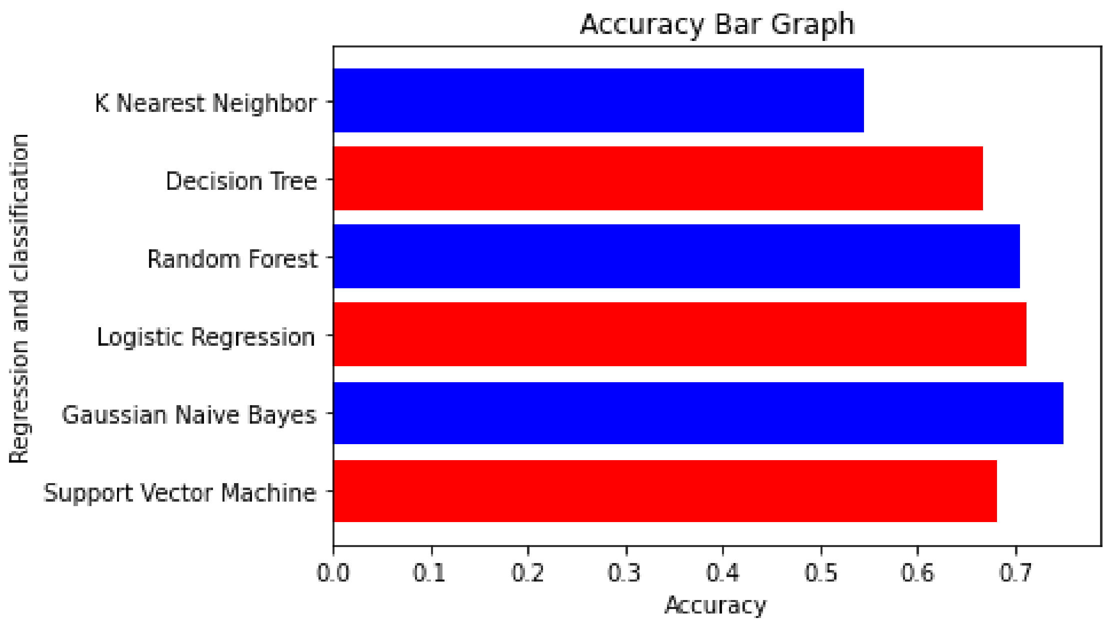

```
Specificity =  0.8611111111111112
```



Accuracy Bar Graph

```
x = ['Support Vector Machine','Gaussian Naive Bayes','Logistic Regression','Random Forest','Decision Tree','K Nearest Neighbor']
y = [SVM,GaussianNB,logisticReg,RandomF,DecisionT,KNN]
plt.barh(x,y,color=['r','b'])
plt.xlabel('Accuracy')
```

```
plt.ylabel('Regression and classification')
plt.title('Accuracy Bar Graph')
plt.show()
```



Thank you