Data Processing

In [230…
```
# importing necessary modules
import pandas as pd
import numpy as np
```

In [231…
```
# TODO: please enter your data file (e.g. "features.csv") path here
df = pd.read_csv("group9_man_2.txt")
# You can also use a full path like below
#df = pd.read_csv("C:/Users/userName/Documents/Python/Coursework_co
```

In [232…
```
display(df)
```

	depArr	distance	angle_error	distance_long	operation_mode	angl
0	0	1061.560659	2	756.386082	5	
1	0	1949.339075	2	866.249197	6	
2	0	918.853912	0	623.358156	5	
3	0	2081.118511	0	1417.437268	5	
4	1	1606.841859	0	849.617019	5	
...	...	...	...	...	...	
1697	0	1612.045476	0	1169.957521	5	
1698	0	2812.914269	0	1417.437268	5	
1699	0	1721.316289	0	1180.036591	6	
1700	0	1525.520928	2	629.022977	6	
1701	0	1725.009843	4	1074.081608	5	

1702 rows × 37 columns

In [233…
```
# deleting columns with NaN
df = df.drop('isSnow', axis=1)
df = df.drop('isFog', axis=1)
df = df.drop('isHail', axis=1)
df = df.drop('budget', axis=1)

display(df)
```

	depArr	distance	angle_error	distance_long	operation_mode	angl
0	0	1061.560659	2	756.386082	5	
1	0	1949.339075	2	866.249197	6	
2	0	918.853912	0	623.358156	5	
3	0	2081.118511	0	1417.437268	5	
4	1	1606.841859	0	849.617019	5	
...	...	...	...	...	...	
1697	0	1612.045476	0	1169.957521	5	
1698	0	2812.914269	0	1417.437268	5	
1699	0	1721.316289	0	1180.036591	6	
1700	0	1525.520928	2	629.022977	6	
1701	0	1725.009843	4	1074.081608	5	

1702 rows × 33 columns

In [234…

```
# deleting columns with NaN - ONLY for MAN and ZRH, skip for HKG
df = df.drop('flightNumber', axis=1)
df = df.drop('airline', axis=1)
df = df.drop('aircraftModel', axis=1)

display(df)
```

	depArr	distance	angle_error	distance_long	operation_mode	angl
0	0	1061.560659	2	756.386082	5	
1	0	1949.339075	2	866.249197	6	
2	0	918.853912	0	623.358156	5	
3	0	2081.118511	0	1417.437268	5	
4	1	1606.841859	0	849.617019	5	
...	...	...	...	...	...	
1697	0	1612.045476	0	1169.957521	5	
1698	0	2812.914269	0	1417.437268	5	
1699	0	1721.316289	0	1180.036591	6	
1700	0	1525.520928	2	629.022977	6	
1701	0	1725.009843	4	1074.081608	5	

1702 rows × 30 columns

In [235…

```
# last column (taxi time) is not included - it is the output
```

```
features = list(df)
print(features)
features = features[:29]
print(features)
```

```
['depArr', 'distance', 'angle_error', 'distance_long', 'operation_mo
de', 'angle_sum', 'QDepDep', 'QDepArr', 'QArrDep', 'QArrArr', 'NDepD
ep', 'NDepArr', 'NArrDep', 'NArrArr', 'Pressure', 'VisibilityInMeter
s', 'TemperatureInCelsius', 'WindSpeedInMPS', 'isRain', 'isDrizzle',
'isMist', 'isHaze', 'aircraft_weight', 'AvgSpdLast5Dep', 'AvgSpdLast
5Arr', 'AvgSpdLast5', 'AvgSpdLast10Dep', 'AvgSpdLast10Arr', 'AvgSpdL
ast10', 'TaxiTime']
['depArr', 'distance', 'angle_error', 'distance_long', 'operation_mo
de', 'angle_sum', 'QDepDep', 'QDepArr', 'QArrDep', 'QArrArr', 'NDepD
ep', 'NDepArr', 'NArrDep', 'NArrArr', 'Pressure', 'VisibilityInMeter
s', 'TemperatureInCelsius', 'WindSpeedInMPS', 'isRain', 'isDrizzle',
'isMist', 'isHaze', 'aircraft_weight', 'AvgSpdLast5Dep', 'AvgSpdLast
5Arr', 'AvgSpdLast5', 'AvgSpdLast10Dep', 'AvgSpdLast10Arr', 'AvgSpdL
ast10']
```

Principal Components Analysis (PCA)

- PCA does a projection from the N-dimensional space to K-dimensional space
- It represents the data as accurately as possible in the lower-dimensional space
- PCA seeks a projection that preserves as much information in the data as possible

In [236…

```
from sklearn.preprocessing import StandardScaler

# Separating out the features
x = df.loc[:, features].values

# normalising the features
x = StandardScaler().fit_transform(x)

#print(x)

np.mean(x),np.std(x) # just checking the normalisation process
```

Out[236…

```
(-5.412781487211752e-17, 1.0)
```

Information Loss in PCA

Below cell runs PCA on our dataset.

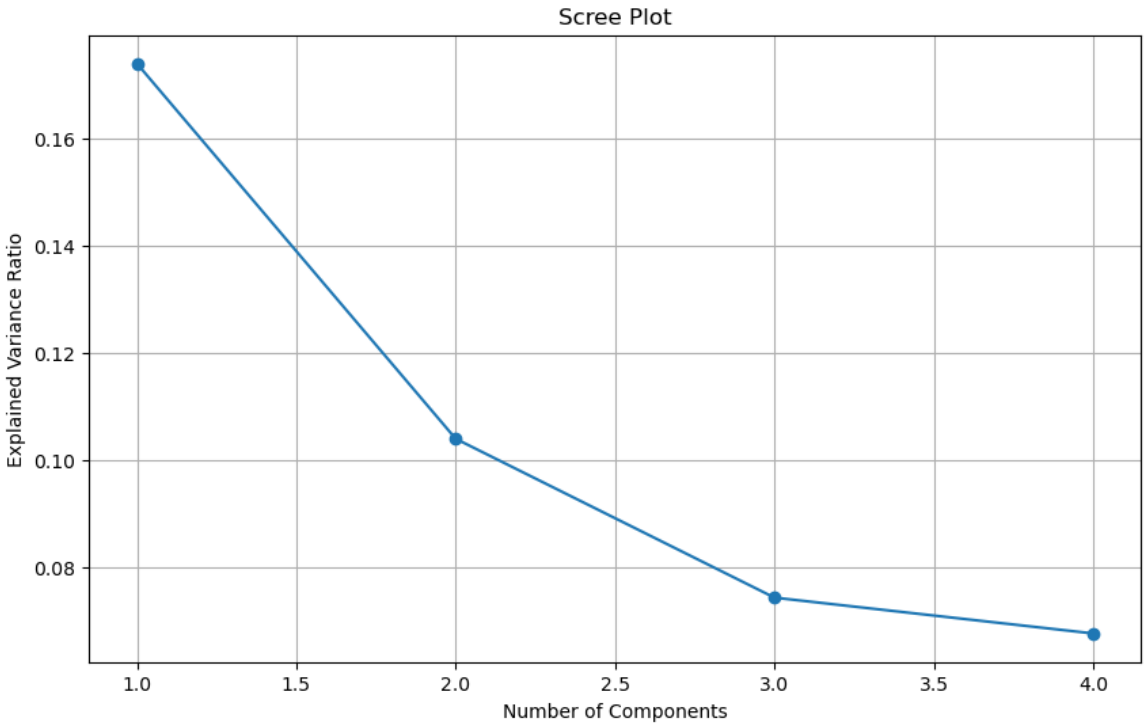
- principalComponents keeps the projected data onto PCs
- explainedVars shows variances: PC1 accounts for 21% of variance, PC2

10%, etc...; so the information loss can be calculated using these variances.

In [237…
```
from sklearn.decomposition import PCA

pca = PCA(n_components=4)
principalComponents = pca.fit_transform(x)
explainedVars = pca.explained_variance_ratio_
print(explainedVars)

sum = 0
for i in explainedVars:
    sum = sum + i

print(sum)
```

```
[0.17406435 0.10397262 0.07429985 0.06760695]
0.4199437762197633
```

In [238…
```
explained_variance_ratio = pca.explained_variance_ratio_
import matplotlib.pyplot as plt
# Step 3: Plot scree plot
plt.figure(figsize=(10, 6))
plt.plot(np.arange(1, len(explained_variance_ratio) + 1), explained_
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance Ratio')
plt.title('Scree Plot')
plt.grid(True)
plt.show()
```



In [239…
```
principalDf = pd.DataFrame(data = principalComponents
                , columns = ['principal component 1', 'principal compo
```

In [240…
```
finalDf = pd.concat([principalDf, df[['TaxiTime']]], axis = 1)
```

In [241…

```
training = df.sample(frac = 0.75)
testing = df.drop(training.index)
```

- For the importance of features we look at pca.components_
- columns correspond to PCs, rows to variables
- we look at the 1st row and find the largest absolute value
- this is our most important variable
- then we look for the 2nd largest, etc.
- in this example the most important are in this order: Feature0, Feature6, Feature12, . . .

In [242…

```
print(abs( pca.components_ ))
```

```
[[0.36373574 0.19584838 0.21051795 0.19788233 0.03571642 0.22327699
  0.34625256 0.19673622 0.19321789 0.15214337 0.38214584 0.07944627
  0.28954599 0.15457105 0.06280553 0.02078432 0.02496306 0.07999708
  0.02534315 0.00057688 0.0102151  0.01949352 0.09458835 0.1390369
  0.13522969 0.22981675 0.15164174 0.16134781 0.23376851]
 [0.12423443 0.20833118 0.08967209 0.24297786 0.19167926 0.10509305
  0.11319259 0.24486045 0.08692247 0.06829474 0.07760752 0.21043604
  0.15404243 0.04335524 0.02427645 0.07217049 0.08498551 0.10009777
  0.00834403 0.00715484 0.04704655 0.09595523 0.10131395 0.27816437
  0.28529504 0.37736228 0.27328051 0.29257965 0.39886467]
 [0.20473357 0.29527391 0.0430445  0.128998   0.22263877 0.26808963
  0.03506548 0.02186396 0.06577993 0.36400426 0.10853302 0.0077549
  0.01032312 0.34691576 0.23374057 0.23626129 0.27496756 0.12768184
  0.23468081 0.00458989 0.08134564 0.11610397 0.19103251 0.25935341
  0.03335666 0.00822687 0.27674479 0.02893768 0.01460936]
 [0.15958998 0.2950811  0.03458344 0.22386293 0.08859876 0.19942891
  0.03053034 0.00848499 0.39783372 0.29782813 0.04413203 0.06519223
  0.41763972 0.39923628 0.08351168 0.17019371 0.04863688 0.01608759
  0.12406405 0.01131333 0.13230061 0.05197854 0.0617052  0.21812425
  0.13002883 0.071242   0.18276044 0.13235487 0.04428282]]
```

In [243…

```
threshold = 0.02

for component in range(4):
    component_loadings = abs(pca.components_[component])
    features_to_drop = [features[i] for i in range(len(features)) i
    features_to_drop = [feature for feature in features_to_drop if
    df = df.drop(features_to_drop, axis=1)
    print("Features dropped for component", component+1, ":", featu

print("\nUpdated DataFrame:")
display(df)
```

```
Features dropped for component 1 : ['isDrizzle', 'isMist', 'isHaze']
Features dropped for component 2 : ['isRain']
Features dropped for component 3 : ['NDepArr', 'NArrDep', 'AvgSpdLas
t5', 'AvgSpdLast10']
Features dropped for component 4 : ['QDepArr', 'WindSpeedInMPS']

Updated DataFrame:
```

	depArr	distance	angle_error	distance_long	operation_mode	angl
0	0	1061.560659	2	756.386082	5	
1	0	1949.339075	2	866.249197	6	
2	0	918.853912	0	623.358156	5	
3	0	2081.118511	0	1417.437268	5	
4	1	1606.841859	0	849.617019	5	
...	...	...	...	...	...	
1697	0	1612.045476	0	1169.957521	5	
1698	0	2812.914269	0	1417.437268	5	
1699	0	1721.316289	0	1180.036591	6	
1700	0	1525.520928	2	629.022977	6	
1701	0	1725.009843	4	1074.081608	5	

1702 rows × 20 columns

In [ ]:

In [244…

```
finalDf.to_csv("data_airport.csv", header=False, index=False)
trainDf = finalDf[:int(76*len(finalDf)/100)]
testDf = finalDf[int(76*len(finalDf)/100):]
print(trainDf)
print(testDf)
```

```
      principal component 1  principal component 2  principal compon
ent 3  \
0                 -0.240190               0.384616               0.3
71833
1                  0.425526               3.331509              -0.3
10274
2                 -0.522303              -0.364934              -1.1
48108
3                 -1.697518              -1.906344              -0.5
99323
4                  2.370078              -1.213871              -0.6
17265
...                     ...                    ...
...
1288              -3.950525               0.224036               1.2
46278
1289              -3.887851               1.673694               0.9
72885
1290               0.195024               4.835288              -0.7
11195
1291              -3.337933               0.566413               2.4
18709
1292              -0.650621               0.743081              -1.2
84055
```

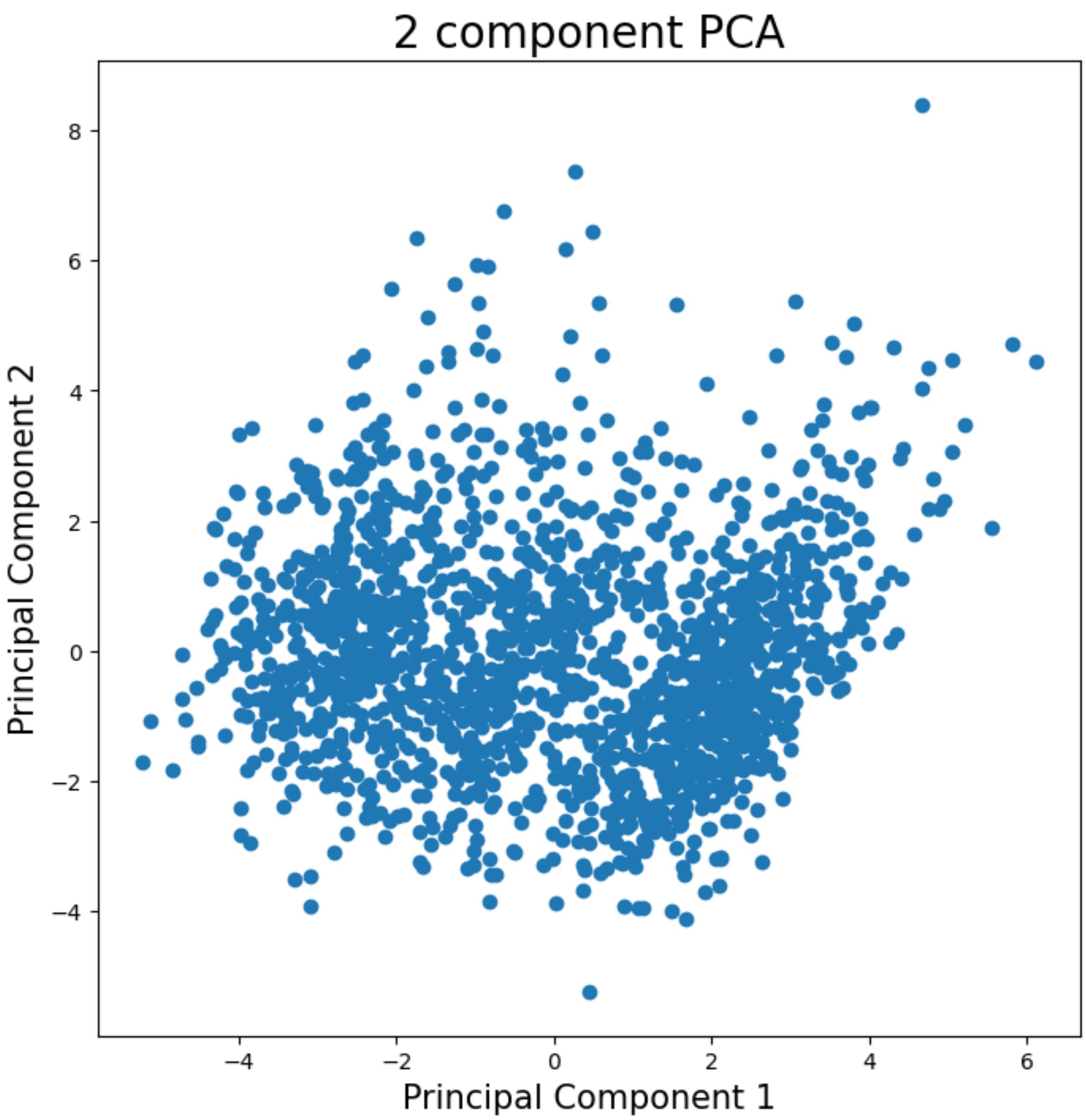
```
      principal component 4   TaxiTime
0                  0.397173   7.700000
1                 -0.782291   7.687667
2                  1.405974   7.666667
3                  0.481950   7.666667
4                 -2.099149   7.666667
...                     ...        ...
1288               0.076338  17.083333
1289              -0.444626  17.083333
1290               1.797714  17.083333
1291              -1.011858  17.066667
1292               1.339715  16.983333

[1293 rows x 5 columns]
      principal component 1  principal component 2  principal compon
ent 3  \
1293              -2.925768               0.548938                0.5
81135
1294              -1.788129               4.025801                0.3
64201
1295              -2.057763               5.576156                1.0
96002
1296              -2.965048              -1.227062               -0.5
93482
1297              -2.745493              -0.876397                0.1
20225
...                     ...                    ...
...
1697              -1.544547              -0.575210               -0.9
55339
1698              -3.488093              -0.847298                1.2
95924
1699              -1.677710              -0.851297               -2.6
14895
1700              -2.509102              -1.295990               -0.5
09787
1701              -2.457001               0.858041                1.1
49787

      principal component 4   TaxiTime
1293              -1.142929  16.983333
1294              -1.366367  16.973167
1295              -3.138806  16.966667
1296               0.528905  16.939683
1297              -0.045641  16.900467
...                     ...        ...
1697               0.063396  13.083333
1698              -1.193735  13.083333
1699               0.377288  13.079933
1700               0.418439  13.066667
1701              -0.593531  13.066667

[409 rows x 5 columns]
```

In [229…

```
import matplotlib.pyplot as plt
```

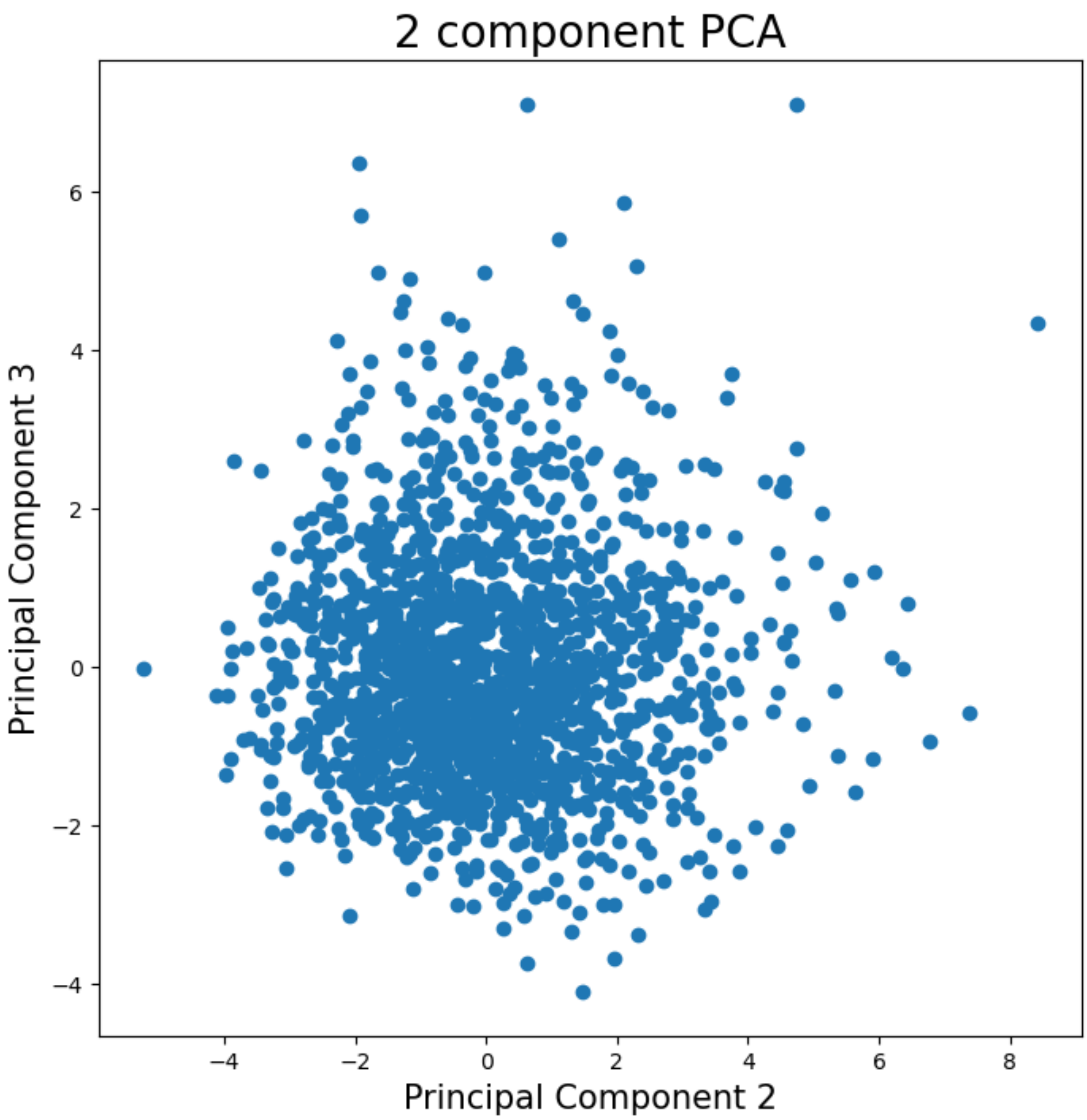
```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)

ax.scatter(finalDf.loc[:, 'principal component 1'], finalDf.loc[:,
```

Out[229… <matplotlib.collections.PathCollection at 0x7ff4b00e60b0>



In [212…
```
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 2', fontsize = 15)
ax.set_ylabel('Principal Component 3', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)

ax.scatter(finalDf.loc[:, 'principal component 2'], finalDf.loc[:,
```

Out[212… <matplotlib.collections.PathCollection at 0x7ff4b0563b20>

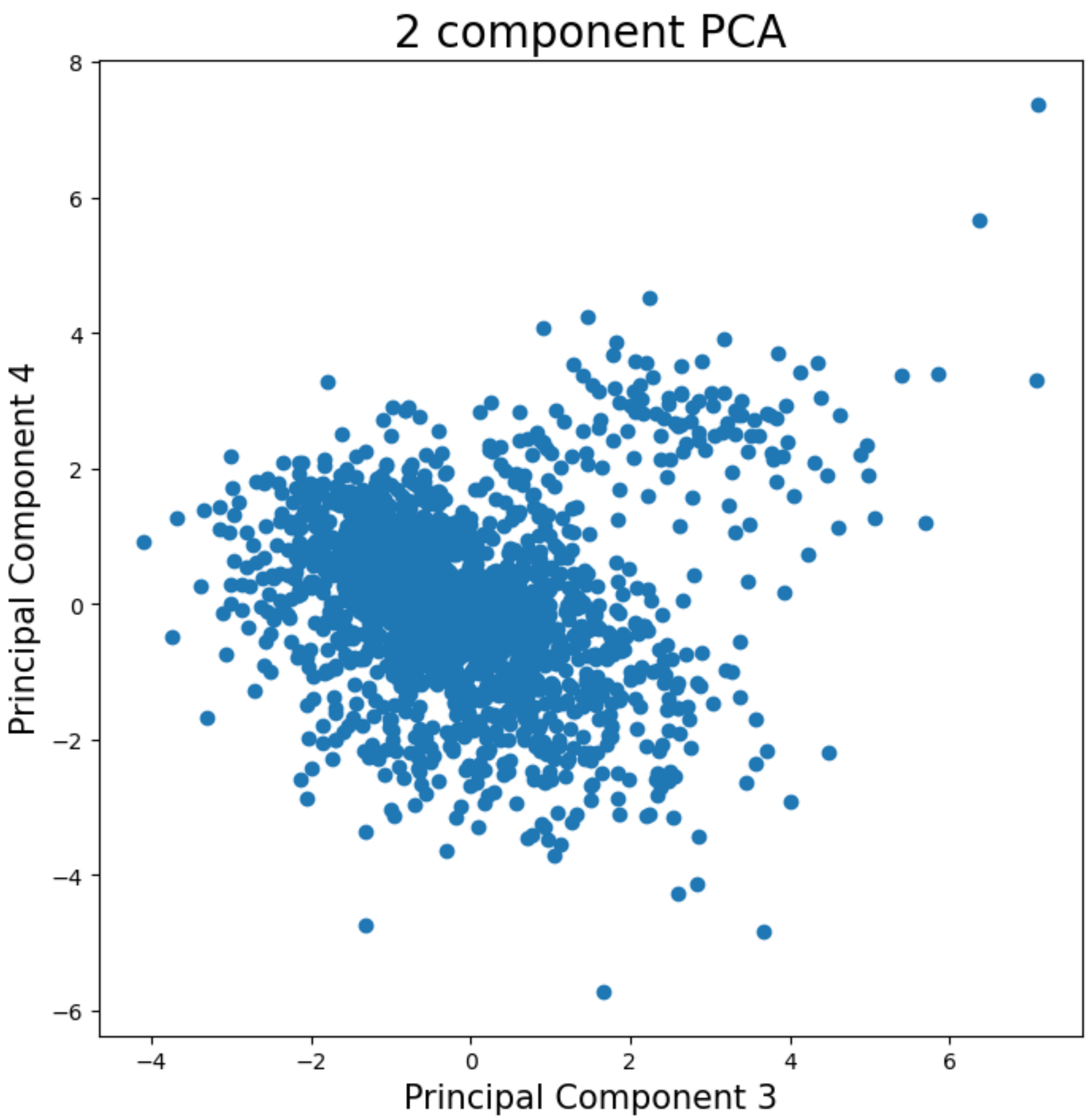


In [213…

```
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 3', fontsize = 15)
ax.set_ylabel('Principal Component 4', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)

ax.scatter(finalDf.loc[:, 'principal component 3'], finalDf.loc[:,
```

Out[213…

```
<matplotlib.collections.PathCollection at 0x7ff4b0572d10>
```

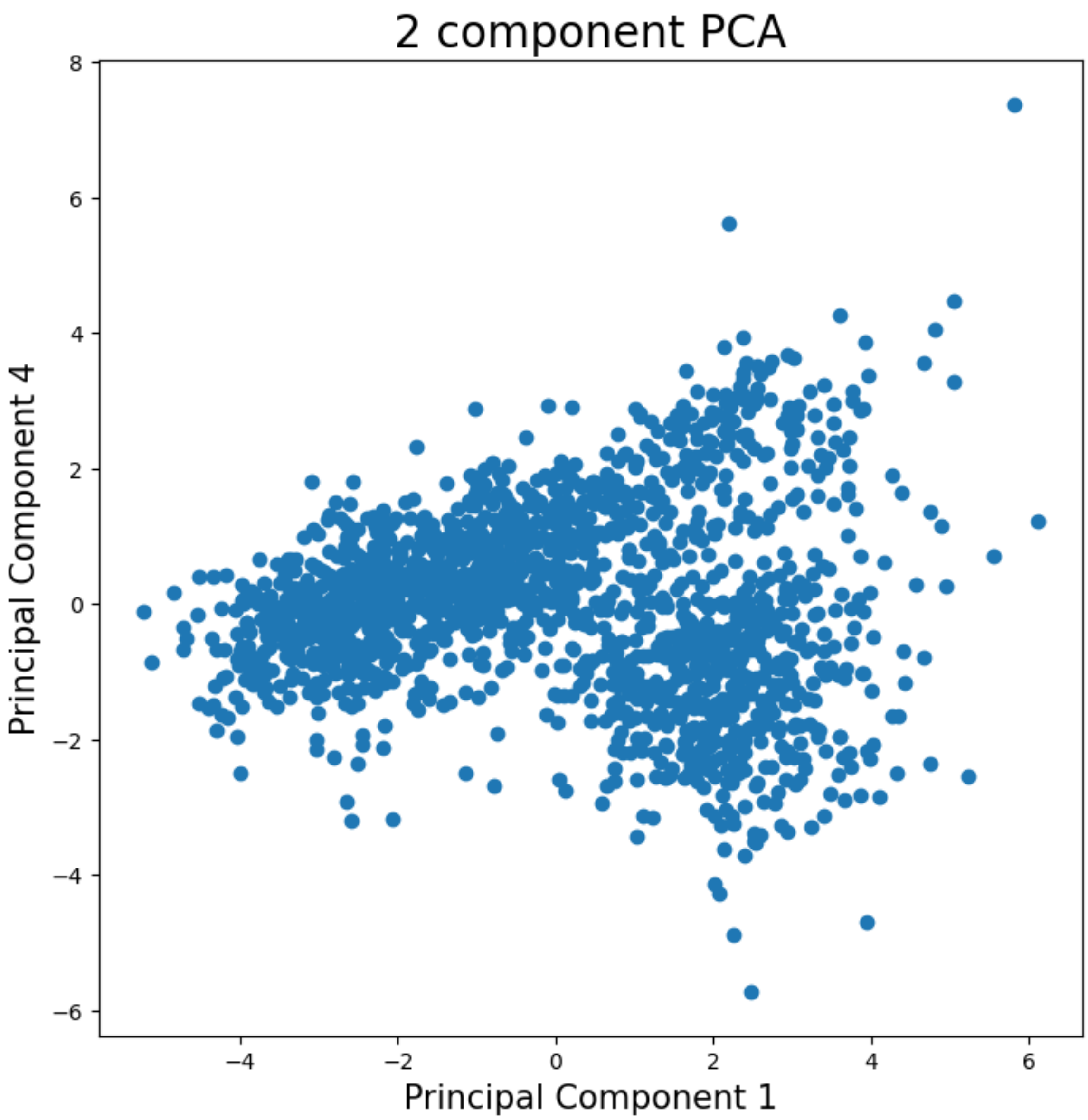


In [178…

```
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 4', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)

ax.scatter(finalDf.loc[:, 'principal component 1'], finalDf.loc[:,
```

Out[178…

```
<matplotlib.collections.PathCollection at 0x7ff4b08cbca0>
```

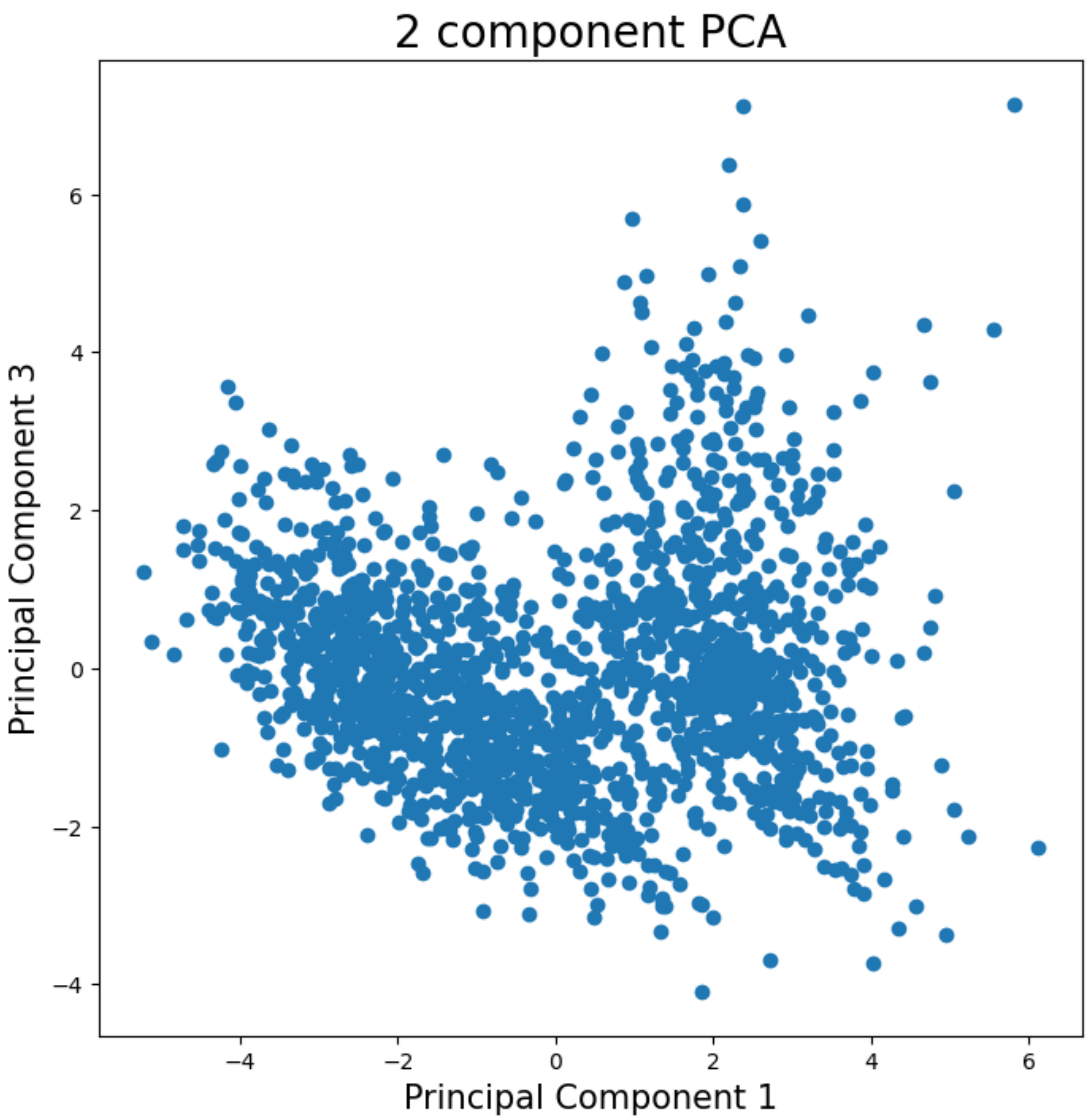


In [179…

```
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 3', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)

ax.scatter(finalDf.loc[:, 'principal component 1'], finalDf.loc[:,
```

Out[179…

```
<matplotlib.collections.PathCollection at 0x7ff4b083afe0>
```



In [180…
```
finalDf.to_csv('data_airport.csv')
finalDf=finalDf.sample(frac=1).reset_index(drop=True)
trainDf=finalDf[:int(74*len(finalDf)/100)]
testDf=finalDf[int(74*len(finalDf)/100):]
print(trainDf)
print(testDf)

trainDf.to_csv('train_data.csv', header=False, index=False)
testDf.to_csv('test_data.csv', header=False, index=False)
```

```
     principal component 1  principal component 2  principal compon
ent 3  \
0                -0.087371               2.331776              -1.8
54450
1                 1.984851              -1.877715              -1.4
71456
2                -1.574539              -2.971624              -0.1
88868
3                -0.125236               3.248088              -2.3
86443
4                 0.363486              -0.576551              -1.3
30725
```

```
...                    ...                    ...                    
...
1254             -2.432492              -0.022045                 0.8
30016
1255              0.158640              -0.449928                -0.9
17786
1256             -2.780052               2.703983                 1.7
23938
1257             -0.686603              -0.310742                -0.0
01067
1258              2.605377              -0.668304                 0.7
93617

      principal component 4   TaxiTime
0                  1.382407  18.070300
1                 -0.770610   5.200000
2                  0.078665   3.633333
3                  0.476201  14.304100
4                  1.519335   1.645017
...                     ...        ...
1254              -0.134370   7.304733
1255               0.304049   6.300000
1256              -0.436244  15.133333
1257               1.167431   7.533333
1258              -3.404286   5.982450

[1259 rows x 5 columns]
      principal component 1  principal component 2  principal compon
ent 3  \
1259               2.111625              -0.356522                -0.0
42493
1260               2.839764              -1.177741                -0.4
72005
1261              -2.713061               1.743582                 1.2
85515
1262               1.455899              -2.800918                 0.5
93635
1263               2.042387              -0.017388                -1.1
78730
...                     ...                    ...                    
...
1697              -0.001010              -0.163990                -0.9
69347
1698               1.177659              -0.594218                 0.3
10836
1699               3.092960               1.444116                 2.3
27234
1700               1.515547              -1.253086                 0.3
66065
1701              -3.877109               0.108173                 1.1
86405

      principal component 4   TaxiTime
1259              -1.402156   4.841917
1260               0.419299   3.066667
1261              -0.466364  16.750000
```

```
1262               -1.217892   7.413617
1263               -1.017908   4.421433
...                      ...        ...
1697                1.174551   5.122800
1698               -0.765012   3.526267
1699               -2.587778   4.766667
1700               -0.930916   4.865867
1701               -1.014004  13.516667

[443 rows x 5 columns]
```

Another PCA Example: Breast Cancer

In [45]:

```
from sklearn.datasets import load_breast_cancer
import pandas as pd
import numpy as np

breast = load_breast_cancer()
breast_data = breast.data
breast_labels = breast.target

labels = np.reshape(breast_labels,(569,1))

final_breast_data = np.concatenate([breast_data,labels],axis=1)

breast_dataset = pd.DataFrame(final_breast_data)

features = breast.feature_names
features_labels = np.append(features,'label')

breast_dataset.columns = features_labels
breast_dataset.head()
```

Out[45]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

5 rows × 31 columns

In [46]:

```
breast_dataset['label'].replace(0, 'Benign',inplace=True)
breast_dataset['label'].replace(1, 'Malignant',inplace=True)

breast_dataset.tail()
```

Out[46]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mea concavi
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.2439
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.1440
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.092
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.3514
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.0000

5 rows × 31 columns

In [47]:
```
from sklearn.preprocessing import StandardScaler

x = breast_dataset.loc[:, features].values
x = StandardScaler().fit_transform(x) # normalizing the features

feat_cols = ['feature' + str(i) for i in range(x.shape[1])]
normalised_breast = pd.DataFrame(x,columns=feat_cols)
normalised_breast.tail()
```

Out[47]:

	feature0	feature1	feature2	feature3	feature4	feature5	feat
564	2.110995	0.721473	2.060786	2.343856	1.041842	0.219060	1.94
565	1.704854	2.085134	1.615931	1.723842	0.102458	-0.017833	0.69
566	0.702284	2.045574	0.672676	0.577953	-0.840484	-0.038680	0.04
567	1.838341	2.336457	1.982524	1.735218	1.525767	3.272144	3.29
568	-1.808401	1.221792	-1.814389	-1.347789	-3.112085	-1.150752	-1.11

5 rows × 30 columns

In [48]:
```
from sklearn.decomposition import PCA

pca_breast = PCA(n_components=2)
principalComponents_breast = pca_breast.fit_transform(x)

principal_breast_Df = pd.DataFrame(data = principalComponents_breas
                                   columns = ['principal component

principal_breast_Df.tail()
```

Out[48]:

	principal component 1	principal component 2
564	6.439315	-3.576817
565	3.793382	-3.584048
566	1.256179	-1.902297
567	10.374794	1.672010
568	-5.475243	-0.670637

In [49]:

```
print('Explained variation per principal component: {}'.format(pca_
```

```
Explained variation per principal component: [0.44272026 0.18971182]
```

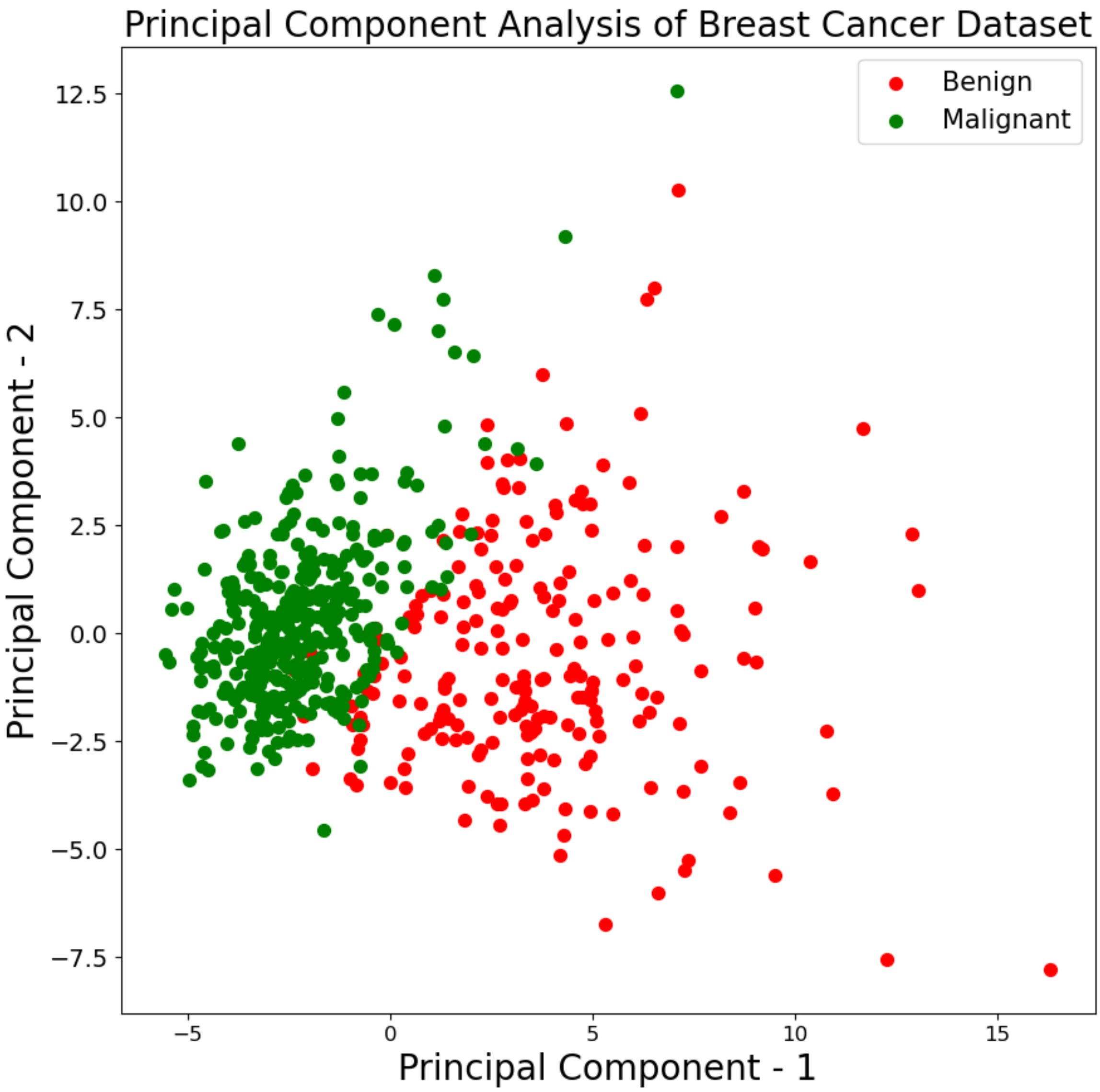
From the above output, you can observe that the principal component 1 holds 44.2% of the information while the principal component 2 holds only 19% of the information. Also, the other point to note is that while projecting thirty-dimensional data to a two-dimensional data, 36.8% information was lost.

Let's plot the visualization of the 569 samples along the principal component - 1 and principal component - 2 axis. It should give you good insight into how your samples are distributed among the two classes.

In [50]:

```
plt.figure()
plt.figure(figsize=(10,10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)
plt.xlabel('Principal Component - 1',fontsize=20)
plt.ylabel('Principal Component - 2',fontsize=20)
plt.title("Principal Component Analysis of Breast Cancer Dataset",f
targets = ['Benign', 'Malignant']
colors = ['r', 'g']

for target, color in zip(targets,colors):
    indicesToKeep = breast_dataset['label'] == target
    plt.scatter(principal_breast_Df.loc[indicesToKeep, 'principal c
                principal_breast_Df.loc[indicesToKeep, 'principal c

plt.legend(targets,prop={'size': 15});
```

```
<Figure size 640x480 with 0 Axes>
```


Principal Component Analysis of Breast Cancer Dataset
Benign
Malignant
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
Principal Component - 2
−5
0
5
10
15
Principal Component - 1

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: