Data Processing and PCA in Python Notebook

June 28, 2025

1 Data Processing

[53]:
```
# importing necessary modules
import pandas as pd
import numpy as np
```

[54]:
```
# TODO: please enter your data file (e.g. "features.csv") path here!
df = pd.read_csv("group15_zrh_2.txt")
# You can also use a full path like below
#df = pd.read_csv("C:/Users/userName/Documents/Python/Coursework_code/features.
↪csv")
```

[55]:
```
display(df)
```

```
      depArr     distance  angle_error  distance_long  operation_mode  \
0          0  1271.509646            0    1154.503523               1
1          1  1343.288525            0       0.000000               1
2          1  1397.409330            0    1195.781351               1
3          1  1959.466717            0    1002.727556               1
4          1  1283.550113            0     705.814166               1
…        …          …          …            …              …
2086       1  1781.973382            0    1370.481831               1
2087       1   943.524755            0     523.162903               1
2088       1   880.930489            0       0.000000               1
2089       1  1461.736387            0     715.741751               1
2090       1  1021.754451            0       0.000000               1

      angle_sum  QDepDep  QDepArr  QArrDep  QArrArr  …  aircraftModel  \
0        184.35        0        1        0        0  …          RJ1H
1        361.32        0        0        1        2  …          A320
2        252.80        0        0        0        0  …          A320
3        371.82        0        0        0        1  …          CL35
4        221.48        0        0        0        1  …          B788
…          …      …      …      …      …  …            …
2086     308.96        0        0        0        1  …          B737
2087     150.94        0        0        2        1  …          A320
2088     165.93        0        0        0        0  …          BCS1
2089     517.66        0        0        0        1  …          E190
```

```
2090     403.23         0         0         0         0  …            E75S

      budget  aircraft_weight  AvgSpdLast5Dep  AvgSpdLast5Arr  AvgSpdLast5  \
0          2                2      923.458513      638.227302   547.451847
1          2                2      772.760777      422.466911   711.614180
2          2                2      719.243355      369.970677   796.434731
3          2                2      272.024961      784.686171   784.686171
4          2                3      191.752999      466.264018   257.482637
…        …              …             …              …           …
2086       2                2      306.482745      711.994356   740.026945
2087       2                2      318.167766      514.274892   547.229076
2088       2                2      143.736008      379.681370   209.529580
2089       2                2      816.008554      589.129418   913.271386
2090       2                2     1039.197565      879.847767  1132.913666

      AvgSpdLast10Dep  AvgSpdLast10Arr  AvgSpdLast10  TaxiTime
0          570.679729       549.984815    647.355269  4.016667
1          510.216071       457.367184    645.252246  4.015017
2          671.550685       510.461577    590.122166  4.014000
3          306.537190       654.471600    628.963405  4.011700
4          168.499462       420.008431    298.249757  4.011450
…               …                …             …        …
2086       282.164119       780.234990    509.238550  2.471217
2087       485.913522       519.609444    471.206767  2.467350
2088       232.846112       378.521971    217.102764  2.460367
2089       590.003413       468.357026    702.568986  2.456500
2090       630.036154       625.752021    959.522666  2.452233

[2091 rows x 37 columns]
```

[56]:
```python
# 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  \
0          0  1271.509646            0    1154.503523               1
1          1  1343.288525            0       0.000000               1
2          1  1397.409330            0    1195.781351               1
3          1  1959.466717            0    1002.727556               1
4          1  1283.550113            0     705.814166               1
…        …          …          …            …              …
2086       1  1781.973382            0    1370.481831               1
2087       1   943.524755            0     523.162903               1
2088       1   880.930489            0       0.000000               1
```

```
2089        1   1461.736387            0      715.741751                 1
2090        1   1021.754451            0        0.000000                 1

      angle_sum  QDepDep  QDepArr  QArrDep  QArrArr  …  airline  \
0        184.35        0        1        0        0  …      SWR
1        361.32        0        0        1        2  …      VLG
2        252.80        0        0        0        0  …      VLG
3        371.82        0        0        0        1  …      NaN
4        221.48        0        0        0        1  …      OMA
…           …      …       …       …       …   …      …
2086     308.96        0        0        0        1  …      SAS
2087     150.94        0        0        2        1  …      SWR
2088     165.93        0        0        0        0  …      SWR
2089     517.66        0        0        0        1  …      OAW
2090     403.23        0        0        0        0  …      AZA

      aircraftModel  aircraft_weight  AvgSpdLast5Dep  AvgSpdLast5Arr  \
0              RJ1H                2      923.458513      638.227302
1              A320                2      772.760777      422.466911
2              A320                2      719.243355      369.970677
3              CL35                2      272.024961      784.686171
4              B788                3      191.752999      466.264018
…               …              …             …               …
2086           B737                2      306.482745      711.994356
2087           A320                2      318.167766      514.274892
2088           BCS1                2      143.736008      379.681370
2089           E190                2      816.008554      589.129418
2090           E75S                2     1039.197565      879.847767

      AvgSpdLast5  AvgSpdLast10Dep  AvgSpdLast10Arr  AvgSpdLast10  TaxiTime
0      547.451847       570.679729       549.984815    647.355269  4.016667
1      711.614180       510.216071       457.367184    645.252246  4.015017
2      796.434731       671.550685       510.461577    590.122166  4.014000
3      784.686171       306.537190       654.471600    628.963405  4.011700
4      257.482637       168.499462       420.008431    298.249757  4.011450
…             …               …               …             …       …
2086   740.026945       282.164119       780.234990    509.238550  2.471217
2087   547.229076       485.913522       519.609444    471.206767  2.467350
2088   209.529580       232.846112       378.521971    217.102764  2.460367
2089   913.271386       590.003413       468.357026    702.568986  2.456500
2090  1132.913666       630.036154       625.752021    959.522666  2.452233

[2091 rows x 33 columns]
```

[57]:
```
# deleting columns with NaN - ONLY for MAN and ZRH, skip for HKG
df = df.drop('flightNumber', axis=1)
df = df.drop('airline', axis=1)
```

```python
df = df.drop('aircraftModel', axis=1)

display(df)
```

```
      depArr     distance  angle_error  distance_long  operation_mode  \
0          0  1271.509646            0    1154.503523               1
1          1  1343.288525            0       0.000000               1
2          1  1397.409330            0    1195.781351               1
3          1  1959.466717            0    1002.727556               1
4          1  1283.550113            0     705.814166               1
…        …          …           …            …              …
2086       1  1781.973382            0    1370.481831               1
2087       1   943.524755            0     523.162903               1
2088       1   880.930489            0       0.000000               1
2089       1  1461.736387            0     715.741751               1
2090       1  1021.754451            0       0.000000               1

      angle_sum  QDepDep  QDepArr  QArrDep  QArrArr  …  isMist  isHaze  \
0        184.35        0        1        0        0  …       0       0
1        361.32        0        0        1        2  …       0       0
2        252.80        0        0        0        0  …       0       0
3        371.82        0        0        0        1  …       0       0
4        221.48        0        0        0        1  …       0       0
…           …      …      …      …      …  …     …     …
2086     308.96        0        0        0        1  …       0       0
2087     150.94        0        0        2        1  …       0       0
2088     165.93        0        0        0        0  …       0       0
2089     517.66        0        0        0        1  …       0       0
2090     403.23        0        0        0        0  …       0       0

      aircraft_weight  AvgSpdLast5Dep  AvgSpdLast5Arr  AvgSpdLast5  \
0                   2      923.458513      638.227302   547.451847
1                   2      772.760777      422.466911   711.614180
2                   2      719.243355      369.970677   796.434731
3                   2      272.024961      784.686171   784.686171
4                   3      191.752999      466.264018   257.482637
…                 …             …             …            …
2086                2      306.482745      711.994356   740.026945
2087                2      318.167766      514.274892   547.229076
2088                2      143.736008      379.681370   209.529580
2089                2      816.008554      589.129418   913.271386
2090                2     1039.197565      879.847767  1132.913666

      AvgSpdLast10Dep  AvgSpdLast10Arr  AvgSpdLast10  TaxiTime
0          570.679729       549.984815    647.355269  4.016667
1          510.216071       457.367184    645.252246  4.015017
2          671.550685       510.461577    590.122166  4.014000
```

```
3          306.537190       654.471600    628.963405  4.011700
4          168.499462       420.008431    298.249757  4.011450
…               …              …             …          …
2086       282.164119       780.234990    509.238550  2.471217
2087       485.913522       519.609444    471.206767  2.467350
2088       232.846112       378.521971    217.102764  2.460367
2089       590.003413       468.357026    702.568986  2.456500
2090       630.036154       625.752021    959.522666  2.452233

[2091 rows x 30 columns]
```

[58]:

```python
# last column (taxi time) is not included - it is the output
features = list(df)
print(features)
len(features)
print(features)
features = features[:len(features)-1]
print(features)
len(features)
```

```
['depArr', 'distance', 'angle_error', 'distance_long', 'operation_mode',
'angle_sum', 'QDepDep', 'QDepArr', 'QArrDep', 'QArrArr', 'NDepDep', 'NDepArr',
'NArrDep', 'NArrArr', 'Pressure', 'VisibilityInMeters', 'TemperatureInCelsius',
'WindSpeedInMPS', 'isRain', 'isDrizzle', 'isMist', 'isHaze', 'aircraft_weight',
'AvgSpdLast5Dep', 'AvgSpdLast5Arr', 'AvgSpdLast5', 'AvgSpdLast10Dep',
'AvgSpdLast10Arr', 'AvgSpdLast10', 'TaxiTime']
['depArr', 'distance', 'angle_error', 'distance_long', 'operation_mode',
'angle_sum', 'QDepDep', 'QDepArr', 'QArrDep', 'QArrArr', 'NDepDep', 'NDepArr',
'NArrDep', 'NArrArr', 'Pressure', 'VisibilityInMeters', 'TemperatureInCelsius',
'WindSpeedInMPS', 'isRain', 'isDrizzle', 'isMist', 'isHaze', 'aircraft_weight',
'AvgSpdLast5Dep', 'AvgSpdLast5Arr', 'AvgSpdLast5', 'AvgSpdLast10Dep',
'AvgSpdLast10Arr', 'AvgSpdLast10', 'TaxiTime']
['depArr', 'distance', 'angle_error', 'distance_long', 'operation_mode',
'angle_sum', 'QDepDep', 'QDepArr', 'QArrDep', 'QArrArr', 'NDepDep', 'NDepArr',
'NArrDep', 'NArrArr', 'Pressure', 'VisibilityInMeters', 'TemperatureInCelsius',
'WindSpeedInMPS', 'isRain', 'isDrizzle', 'isMist', 'isHaze', 'aircraft_weight',
'AvgSpdLast5Dep', 'AvgSpdLast5Arr', 'AvgSpdLast5', 'AvgSpdLast10Dep',
'AvgSpdLast10Arr', 'AvgSpdLast10']
```

[58]: 29

2 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

[59]:
```python
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
```

[59]: (-1.0077124503268296e-16, 0.982607368881035)

3 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.

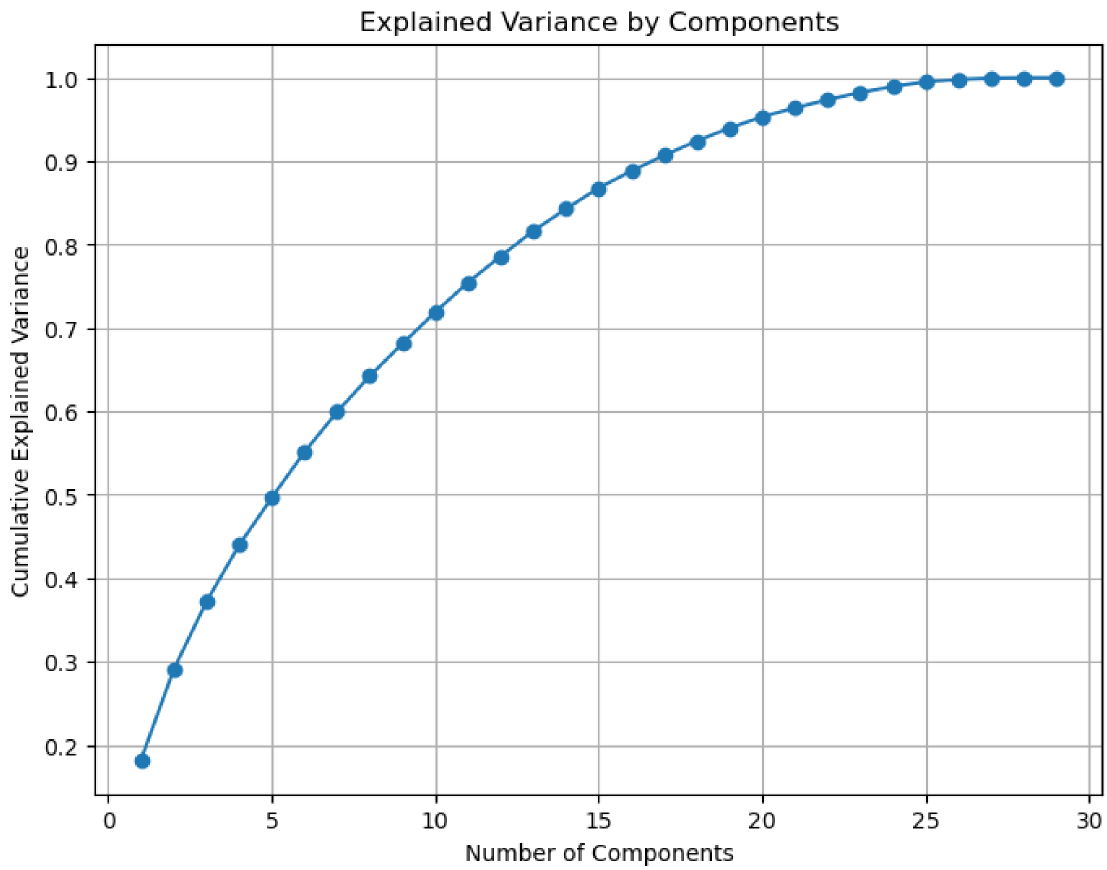
[60]:
```python
from sklearn.decomposition import PCA

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)
```

```
[0.18211975 0.10846973 0.08120562 0.06781795]
0.43961304588779637
```

[61]:
```python
##how many principal components needed
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Assuming 'data' is your DataFrame and 'features' are your features
# Make sure to preprocess your data before this step

# Standardize the features
scaler = StandardScaler()
```

```
data_scaled = scaler.fit_transform(df[features])

#perform PCS
pca = PCA()
data_pca = pca.fit_transform(data_scaled)

#calculate explained variance
explained_variance = np.cumsum(pca.explained_variance_ratio_)
print(explained_variance)
#plot the explained variance
plt.figure(figsize=(8,6))
plt.plot(range(1, len(explained_variance)+1), explained_variance, marker='o')
plt.title('Explained Variance by Components')
plt.xlabel('Number of Components')
plt.ylabel ('Cumulative Explained Variance')
plt.grid(True)
plt.show()
```

```
[0.18211976 0.29058956 0.37179536 0.43961369 0.49665421 0.55126858
 0.59947654 0.64251009 0.68145369 0.71908699 0.75473456 0.78613758
 0.81655538 0.84285458 0.86798893 0.88820851 0.90722963 0.92445671
 0.93992195 0.95340554 0.96395837 0.97385319 0.98252906 0.98963535
 0.99534705 0.99807226 0.9995587  1.         1.        ]
```




```
[62]: principalDf = pd.DataFrame(data = principalComponents
                   , columns = ['principal component 1', 'principal component 2',␣
       ↪'principal component 3', 'principal component 4'])
```

```
[63]: finalDf = pd.concat([principalDf, df[['TaxiTime']]], axis = 1)
```

- 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, . . .

```
[64]: print(abs( pca.components_ ))
```

```
[[4.07181280e-01 1.71943873e-01 1.88788968e-01 2.10019804e-01
  2.65753231e-02 1.91669025e-01 3.51421665e-01 3.06474281e-01
  2.07135714e-01 2.28085074e-01 3.80409653e-01 2.05756866e-01
  3.18520451e-01 2.40675537e-01 4.95807800e-02 2.27799793e-02
  9.02289551e-04 4.49421336e-02 3.15848225e-02 2.16347634e-02
```

```
 2.14479343e-02 7.01700538e-32 5.17866163e-02 7.88371898e-02
 1.50005475e-02 6.39823179e-02 5.92259328e-02 1.38297045e-02
 6.94869982e-02]
[2.46078021e-02 9.45773071e-02 6.16698377e-03 9.31491654e-02
 4.88470185e-02 7.85048272e-02 2.14000361e-03 4.09458613e-02
 8.06455687e-03 8.11372788e-03 1.37114011e-02 8.21307052e-02
 4.13782264e-02 2.79328470e-02 9.78205630e-02 8.03565833e-03
 8.42418700e-02 6.60741589e-02 8.28365452e-02 1.19395514e-03
 2.85187852e-03 2.64804829e-26 1.50948127e-02 3.35666697e-01
 3.81260882e-01 4.52450451e-01 2.97454591e-01 3.42609858e-01
 5.14925726e-01]
[1.44032953e-01 5.21504166e-01 3.79324394e-02 4.38506755e-01
 7.90184867e-02 3.57008513e-01 4.33073033e-02 3.02593315e-02
 2.99907464e-01 2.56022812e-01 4.53106056e-02 1.62552787e-01
 2.00612400e-01 9.00439824e-02 7.23594365e-02 4.52453500e-02
 9.99500576e-02 8.65935659e-02 1.32027519e-01 1.61844985e-02
 6.67436843e-03 1.62851183e-24 9.20072768e-02 1.91768821e-01
 8.31093124e-02 7.12061906e-02 1.67902876e-01 8.77488258e-02
 9.55524327e-02]
[1.40466402e-02 1.64289954e-01 1.86230978e-02 1.27856694e-01
 2.28593055e-01 1.01541027e-01 1.33457592e-03 5.65536555e-02
 5.35018695e-02 1.16706158e-01 3.47726415e-02 4.02216815e-02
 1.20349927e-04 7.68072227e-02 1.33772980e-01 2.44720774e-01
 3.80944774e-01 2.74122072e-01 5.02844965e-02 2.33808637e-02
 1.81290969e-01 5.29395592e-22 8.33734141e-02 3.54596923e-01
 3.52931250e-01 9.49611861e-03 3.86709023e-01 3.52474709e-01
 5.73404692e-03]
[1.23702511e-02 7.25012340e-02 8.14406530e-02 8.50441241e-02
 1.61216572e-01 3.30327587e-02 7.58253993e-03 8.17585044e-02
 1.13326264e-01 1.65336853e-01 4.02038409e-02 1.27567363e-01
 2.10972888e-01 2.93286808e-01 1.64907140e-01 4.86788328e-01
 2.81276924e-01 1.24725925e-01 1.17708357e-01 2.11318683e-01
 3.60160464e-01 0.00000000e+00 6.77122561e-03 2.04779758e-01
 2.77469849e-01 1.47327029e-02 1.63787123e-01 2.55227563e-01
 9.72386486e-03]
[3.00438007e-03 1.33516230e-01 2.88196343e-03 1.06532071e-01
 8.98083185e-02 9.25810501e-02 5.21077211e-02 1.09393563e-01
 5.63893083e-02 1.07310216e-01 1.36787828e-02 4.92623391e-02
 2.30829250e-02 7.53943755e-02 4.31199780e-01 3.06512867e-01
 2.84890556e-02 5.64050087e-01 5.17580109e-01 1.53322468e-01
 8.63055108e-02 4.33680869e-19 5.90789971e-02 5.94860659e-02
 1.72536134e-02 7.47267696e-02 3.34954135e-02 4.03459682e-02
 5.10710532e-02]
[1.41657147e-02 7.34059627e-02 1.55913712e-02 8.31789154e-02
 4.75135577e-01 3.20075459e-02 1.66920325e-02 7.40232343e-02
 1.27812401e-01 2.21929365e-01 5.02615036e-02 1.23143806e-01
 2.22048061e-01 2.97077723e-01 2.89665946e-01 2.90808237e-01
 2.27772545e-01 8.47899939e-02 1.63454415e-01 2.24037276e-01
```

```
 3.31162447e-01 0.00000000e+00 3.13385137e-01 6.11436716e-02
 7.51997552e-02 1.95291390e-02 8.14878065e-02 6.89024573e-02
 1.93869786e-02]
[1.67210222e-02 5.82430873e-02 1.30969877e-01 3.79568279e-02
 1.61320344e-01 4.97975141e-02 1.61322798e-01 2.45039069e-01
 2.44335192e-01 2.49433612e-01 2.64018146e-01 4.79457240e-01
 3.19990849e-01 4.30116564e-01 9.59641791e-02 4.24365008e-02
 1.08852879e-01 5.81520263e-02 7.13167262e-02 1.31397985e-01
 1.24459468e-01 3.46944695e-18 2.40498239e-01 5.78184850e-02
 1.03724664e-01 2.61534683e-02 5.34632559e-02 1.17852319e-01
 2.84624569e-02]
[2.91629081e-02 2.37517047e-02 2.08997835e-01 8.23427598e-03
 3.33984873e-02 4.07830745e-03 8.91267423e-02 2.41597984e-01
 9.01698682e-03 1.73502274e-01 1.49471694e-01 3.53191340e-01
 9.79788293e-02 1.28147300e-01 1.90435741e-01 4.07595696e-02
 3.28152208e-01 2.24245938e-01 1.52074724e-01 5.75872144e-01
 1.38340940e-01 2.77555756e-17 1.21064041e-01 1.80541517e-01
 1.19786896e-01 2.85159151e-02 1.91061887e-01 1.23660307e-01
 9.95932041e-03]
[1.50742084e-02 2.61403476e-02 3.07574949e-01 1.03702110e-01
 2.30210379e-01 2.72904887e-01 5.14365891e-02 1.92058376e-01
 1.00442178e-01 1.07033147e-01 1.73266275e-01 3.18723717e-01
 1.07653926e-01 1.93163654e-01 1.32490853e-01 7.16946739e-02
 1.13189492e-01 3.96221833e-02 1.49425848e-02 3.28483071e-01
 1.16354106e-01 5.55111512e-17 5.80586385e-01 1.19263437e-01
 4.79593885e-02 2.39413159e-02 8.08958222e-02 7.17457892e-02
 8.18637126e-03]
[1.25309745e-02 3.88768974e-02 1.22296298e-01 8.47545831e-02
 2.88893705e-01 2.33794672e-01 4.02499333e-02 1.00581669e-02
 1.08002466e-01 1.00529375e-01 7.50127183e-03 3.36508799e-02
 1.28595338e-01 1.74024658e-01 6.59396469e-02 2.46302586e-02
 2.41449323e-01 1.23050069e-01 2.71110196e-02 4.09375660e-01
 4.78220698e-01 8.32667268e-17 4.93338167e-01 1.05451462e-01
 1.06824978e-01 1.02424131e-01 3.38530069e-02 1.15620327e-01
 1.54671350e-02]
[2.71305045e-02 2.25942475e-01 6.31588370e-01 2.09753646e-01
 2.41220939e-02 5.07448881e-02 9.34428572e-02 1.97562174e-01
 4.07294493e-01 3.27668969e-01 3.06851226e-02 1.59859515e-01
 1.17950074e-02 1.68193381e-03 7.60311145e-02 2.32048855e-02
 2.03042688e-01 5.76806082e-02 8.19468582e-02 1.85945142e-01
 9.95145982e-02 8.32667268e-17 1.45774207e-01 3.21707508e-02
 3.73245068e-02 1.16333933e-01 1.17900062e-01 2.22529849e-02
 2.87367941e-02]
[9.34355087e-03 2.84197956e-02 2.16466030e-01 4.30592483e-02
 8.58599629e-02 1.69208415e-02 7.86670570e-02 7.69770804e-02
 1.14578509e-02 4.10483590e-02 6.50425845e-03 2.33102017e-02
 4.04721809e-03 3.76714587e-03 5.51189929e-01 1.48384963e-01
 1.32475091e-01 8.84555721e-02 6.10036570e-01 2.26059678e-01
```

```
 3.82633887e-01 1.66533454e-16 2.12997350e-02 2.53476731e-02
 5.85724432e-04 5.32392786e-02 1.78209745e-02 4.63560953e-02
 3.05364461e-02]
[6.19047892e-02 2.51582871e-02 4.36422702e-01 2.27054723e-02
 4.94126002e-01 2.41284983e-01 3.10467736e-01 8.58258101e-02
 2.05453135e-01 7.92171631e-02 2.07277369e-01 2.36533354e-01
 3.41218640e-03 9.69813824e-02 4.08440889e-02 1.88613132e-02
 2.40843295e-01 1.72567970e-01 8.41819969e-02 1.19183247e-01
 2.94172301e-01 5.55111512e-17 6.80240136e-02 1.13276654e-01
 3.72119564e-02 1.09431646e-01 8.91432414e-02 6.83730157e-02
 2.29334576e-02]
[1.69056087e-01 8.59788607e-02 2.88849499e-01 1.24653369e-01
 3.98375739e-01 7.71336858e-02 1.86357387e-01 9.86360955e-02
 4.62636964e-01 3.16941718e-01 2.01269758e-01 3.77372214e-02
 1.01927200e-01 5.99461448e-02 2.23480313e-01 7.46908284e-02
 2.93833139e-01 1.98723554e-01 1.02208425e-02 1.50651686e-01
 1.29196027e-01 2.77555756e-17 2.17232505e-01 7.74835451e-03
 1.28266060e-02 4.41848332e-02 1.15052985e-01 1.38274266e-03
 4.57581031e-02]
[2.54566787e-01 6.21994184e-02 2.37210384e-02 2.06086927e-01
 8.77085387e-02 9.71430342e-02 3.63039240e-01 4.53557693e-01
 1.18890378e-01 4.76248317e-02 1.53107840e-01 1.52829493e-01
 1.96457448e-01 1.14230130e-01 1.88045709e-02 1.01932958e-01
 2.72877937e-03 7.22696610e-02 8.47375452e-02 1.34131453e-01
 6.38513649e-02 2.77555756e-17 6.87781199e-02 3.97966286e-02
 4.33086184e-02 3.93885852e-01 3.37496842e-01 2.99295416e-01
 8.22443425e-02]
[7.80053368e-02 7.66095166e-02 1.96372123e-01 4.29781934e-01
 6.65064216e-02 6.25659212e-01 1.12355651e-01 1.10584733e-01
 3.63826985e-02 9.73119508e-02 6.08212546e-02 5.74170411e-02
 3.11574785e-02 3.13206335e-02 9.33571212e-02 5.52207376e-02
 9.26443757e-02 3.34515448e-02 1.12104607e-01 4.84651277e-03
 4.23012322e-03 0.00000000e+00 2.21250625e-01 9.11525200e-03
 8.39595569e-02 3.58268934e-01 2.67185515e-01 1.84704895e-01
 1.68764944e-02]
[2.39791197e-01 7.53286668e-02 1.36579981e-01 1.99372047e-01
 2.46937029e-01 3.55792666e-01 2.67318818e-01 3.46235363e-01
 1.83800373e-01 1.13801942e-01 1.14141494e-01 1.79291670e-01
 1.76039699e-01 1.14643777e-01 1.13546100e-01 1.23219187e-01
 5.14866678e-02 4.50022775e-02 9.65196277e-02 4.47661060e-02
 7.59161595e-02 1.38777878e-17 3.06606426e-01 3.47742210e-02
 1.82330345e-02 3.38917672e-01 2.46680991e-01 2.02675387e-01
 3.60442187e-02]
[6.46506663e-03 2.06252516e-02 1.05654079e-02 6.95619311e-03
 1.28756163e-01 5.68013141e-02 1.20592949e-02 6.25489557e-03
 2.21384577e-02 4.09203063e-03 2.95003114e-02 2.22600686e-02
 3.33224884e-02 3.19105719e-02 3.03699730e-01 6.43191084e-01
 3.39279703e-01 2.15849647e-01 2.67709257e-01 2.48830616e-01
```

```
 3.58096613e-01 5.55111512e-17 2.30253874e-02 5.92412331e-02
 5.40947270e-02 1.58866434e-01 3.54148090e-02 1.10218603e-01
 1.80934612e-02]
[5.95773473e-03 6.41123015e-02 1.35306354e-02 1.24306732e-02
 2.85253245e-02 5.17495735e-03 1.59563756e-02 3.35263092e-02
 4.69501901e-02 2.69861837e-02 7.64005308e-04 3.43348488e-03
 3.32565104e-02 6.35551922e-02 3.49453361e-01 1.87520703e-01
 4.20063693e-01 5.92851275e-01 3.75264285e-01 2.23850666e-01
 2.10239223e-01 5.55111512e-17 1.03108096e-02 4.38455424e-02
 9.63805901e-02 9.09353541e-02 1.08461900e-01 1.75747871e-01
 5.12638379e-02]
[1.73909531e-02 3.75287201e-03 8.95163612e-03 4.09382668e-02
 8.65222426e-02 1.77924869e-03 9.29163124e-03 2.98332478e-02
 1.34246500e-01 1.80751868e-01 1.55856949e-02 3.36132363e-02
 4.59065275e-02 1.13382121e-01 3.18894752e-02 5.52983599e-02
 2.40865498e-02 7.95488259e-02 7.31118910e-02 1.77667843e-02
 2.61832839e-02 8.32667268e-17 5.52821547e-02 5.46621127e-01
 3.64070580e-01 1.34735883e-02 4.34932723e-01 4.99208705e-01
 1.79297621e-01]
[2.39315843e-02 8.15053292e-03 3.05936459e-02 4.07140642e-02
 6.72092133e-03 2.97200483e-02 2.61215680e-02 1.07745592e-02
 4.99719511e-01 6.22114302e-01 2.47287843e-02 6.05886361e-03
 2.60893663e-01 4.56598705e-01 1.13799560e-02 4.13216567e-02
 2.58062574e-02 1.26311738e-02 4.63311038e-03 4.72123773e-03
 3.98758513e-03 1.38777878e-16 2.48369597e-02 7.34013553e-02
 1.47691440e-01 1.45952227e-02 7.10533318e-02 1.94290531e-01
 8.68821854e-02]
[1.84205995e-02 4.25309543e-03 4.01158430e-03 9.45567747e-03
 6.64509271e-02 2.64426026e-02 5.74212241e-03 4.77870376e-02
 5.99875235e-02 5.65063605e-02 6.37438517e-03 2.08321247e-02
 2.49617137e-02 4.57122091e-02 2.95954454e-02 4.26193108e-02
 8.07419349e-02 6.68140239e-02 2.64679758e-02 1.57515773e-02
 4.83496090e-03 1.11022302e-16 1.42534762e-03 3.68298944e-01
 2.84709298e-01 5.43079529e-01 3.98281517e-01 3.08500792e-01
 4.49485577e-01]
[5.86666929e-02 3.84309818e-02 1.85899482e-03 3.30226665e-02
 1.88456586e-02 3.02984442e-02 6.77160792e-01 5.60893798e-01
 8.28212503e-04 2.58987774e-02 2.42959133e-01 3.85055459e-01
 5.00479667e-02 5.26959551e-02 1.49733814e-02 4.90433288e-03
 2.94374384e-03 1.54713357e-02 1.72558419e-02 2.68459966e-02
 1.27431066e-02 1.47451495e-17 3.68489950e-02 1.40855136e-02
 4.02258892e-02 1.78907211e-02 4.06771724e-03 2.97999008e-02
 1.54746893e-02]
[2.85434601e-03 1.51149509e-02 5.86094699e-03 8.99744569e-03
 2.99746336e-02 5.29074620e-04 6.48964694e-02 2.45422462e-02
 1.15420549e-02 1.03403323e-02 2.00167929e-02 2.42345897e-02
 3.55313744e-02 5.20307417e-02 2.52269890e-02 1.70992603e-03
 3.17704539e-02 2.97964191e-02 1.04351684e-02 5.23743613e-03
```

```
 1.03224663e-02 1.11022302e-16 9.23479690e-03 3.79907561e-01
 5.82842343e-01 9.72810850e-02 3.05154597e-02 1.80803139e-01
 6.77313404e-01]
[8.25549202e-02 7.29207461e-01 7.61590969e-02 5.99554813e-01
 2.20147975e-02 2.75786714e-01 7.13866111e-02 5.31230088e-03
 2.05965571e-02 1.98834731e-02 8.88319685e-02 3.01382770e-02
 3.56178240e-02 2.20984615e-02 2.42928540e-04 3.19816302e-04
 2.51164260e-02 2.71123281e-02 2.12837174e-02 1.68830089e-03
 3.21647751e-03 1.73472348e-18 1.61999229e-03 2.39932401e-02
 3.08278473e-03 2.31213315e-03 1.42102417e-02 1.99448369e-02
 1.03261560e-02]
[2.74329528e-01 1.20546151e-01 2.26636897e-02 1.88940231e-02
 2.71546287e-03 2.62088273e-02 2.96920415e-02 1.61015061e-02
 6.37167554e-02 3.75978854e-02 4.55161320e-01 2.23686652e-01
 6.68923893e-01 4.43436333e-01 5.12529350e-03 1.44049484e-02
 1.68673991e-04 5.20412183e-03 3.43269174e-03 4.30783806e-03
 7.58936741e-03 6.50521303e-19 4.48032394e-04 1.61144102e-02
 1.45046866e-03 5.45546767e-04 8.74532621e-03 1.68735076e-04
 6.52844024e-03]
[7.49893246e-01 4.45344264e-04 1.18963518e-02 1.32120108e-02
 2.63519512e-03 2.75795352e-03 3.84391357e-02 3.05355179e-02
 1.31235168e-02 8.15810574e-03 5.69647520e-01 2.95796858e-01
 1.24496234e-01 8.38480650e-02 2.44394264e-03 9.29381230e-04
 1.80085750e-03 1.21540266e-03 1.61414026e-03 1.68348132e-03
 5.66003577e-03 4.06575815e-20 3.87665527e-04 6.10145662e-03
 3.99850450e-03 5.06151298e-04 1.80334276e-03 6.49171266e-03
 2.93008014e-03]
[0.00000000e+00 1.04722092e-17 2.17294517e-17 1.81066051e-17
 4.68715361e-17 2.30143986e-17 4.50643361e-17 4.19904499e-17
 6.29236021e-17 7.64865860e-17 2.66115537e-17 5.11762302e-17
 8.57993454e-18 8.62518388e-18 5.87488513e-17 3.93727155e-17
 2.68343869e-17 3.32521904e-17 3.03911163e-17 3.38157549e-17
 7.68734386e-18 1.00000000e+00 1.41133324e-17 4.43311130e-17
 1.69576612e-16 1.12020330e-16 1.19989754e-17 6.19903279e-17
 1.47840184e-17]]
```

[65]:

```python
threshold = 0.01  # Set your desired threshold here

# Iterate through each principal component
for component in range(4):
    component_loadings = abs(pca.components_[component])  # Get absolute␣
 ↪loadings for the current component

    # Drop features that don't meet the threshold
    features_to_drop = [features[i] for i in range(len(features)) if␣
 ↪component_loadings[i] < threshold]
    #Filter out features that are not present in the DataFrame
```

```python
    features_to_drop = [feature for feature in features_to_drop if feature in df.columns]
    df = df.drop(features_to_drop, axis=1)
    print("Features dropped for component", component+1, ":", features_to_drop)
# Print the updated DataFrame
print("\nUpdated DataFrame:")
display(df)

features = list(df)
print(features)
print(len(features))
```

```
Features dropped for component 1 : ['TemperatureInCelsius', 'isHaze']
Features dropped for component 2 : ['angle_error', 'QDepDep', 'QArrDep',
'QArrArr', 'VisibilityInMeters', 'isDrizzle', 'isMist']
Features dropped for component 3 : []
Features dropped for component 4 : ['NArrDep', 'AvgSpdLast5', 'AvgSpdLast10']

Updated DataFrame:
```

	depArr	distance	distance_long	operation_mode	angle_sum	QDepArr	\
0	0	1271.509646	1154.503523	1	184.35	1	
1	1	1343.288525	0.000000	1	361.32	0	
2	1	1397.409330	1195.781351	1	252.80	0	
3	1	1959.466717	1002.727556	1	371.82	0	
4	1	1283.550113	705.814166	1	221.48	0	
…	…	…	…	…	…	…	
2086	1	1781.973382	1370.481831	1	308.96	0	
2087	1	943.524755	523.162903	1	150.94	0	
2088	1	880.930489	0.000000	1	165.93	0	
2089	1	1461.736387	715.741751	1	517.66	0	
2090	1	1021.754451	0.000000	1	403.23	0	

	NDepDep	NDepArr	NArrArr	Pressure	WindSpeedInMPS	isRain	\
0	2	0	0	30.18	3.0888	0	
1	0	0	1	29.85	4.1184	1	
2	0	0	1	30.00	1.0296	0	
3	0	0	2	30.27	2.0592	0	
4	0	0	0	29.85	8.7516	0	
…	…	…	…	…	…	…	
2086	0	0	2	29.71	4.6332	0	
2087	0	0	2	30.21	5.6628	1	
2088	0	0	0	29.91	7.2072	1	
2089	0	0	1	30.27	0.0000	0	
2090	0	0	1	30.21	0.5148	0	

	aircraft_weight	AvgSpdLast5Dep	AvgSpdLast5Arr	AvgSpdLast10Dep	\
0	2	923.458513	638.227302	570.679729	

```
1                     2      772.760777      422.466911       510.216071
2                     2      719.243355      369.970677       671.550685
3                     2      272.024961      784.686171       306.537190
4                     3      191.752999      466.264018       168.499462
…                 …             …             …              …
2086                  2      306.482745      711.994356       282.164119
2087                  2      318.167766      514.274892       485.913522
2088                  2      143.736008      379.681370       232.846112
2089                  2      816.008554      589.129418       590.003413
2090                  2     1039.197565      879.847767       630.036154

      AvgSpdLast10Arr  TaxiTime
0          549.984815  4.016667
1          457.367184  4.015017
2          510.461577  4.014000
3          654.471600  4.011700
4          420.008431  4.011450
…                 …         …
2086       780.234990  2.471217
2087       519.609444  2.467350
2088       378.521971  2.460367
2089       468.357026  2.456500
2090       625.752021  2.452233

[2091 rows x 18 columns]

['depArr', 'distance', 'distance_long', 'operation_mode', 'angle_sum',
'QDepArr', 'NDepDep', 'NDepArr', 'NArrArr', 'Pressure', 'WindSpeedInMPS',
'isRain', 'aircraft_weight', 'AvgSpdLast5Dep', 'AvgSpdLast5Arr',
'AvgSpdLast10Dep', 'AvgSpdLast10Arr', 'TaxiTime']
18
```

[66]:
```python
features = list(df)
print(features)
len(features)
print(features)
features = features[:len(features)-1]
print(features)
len(features)
```

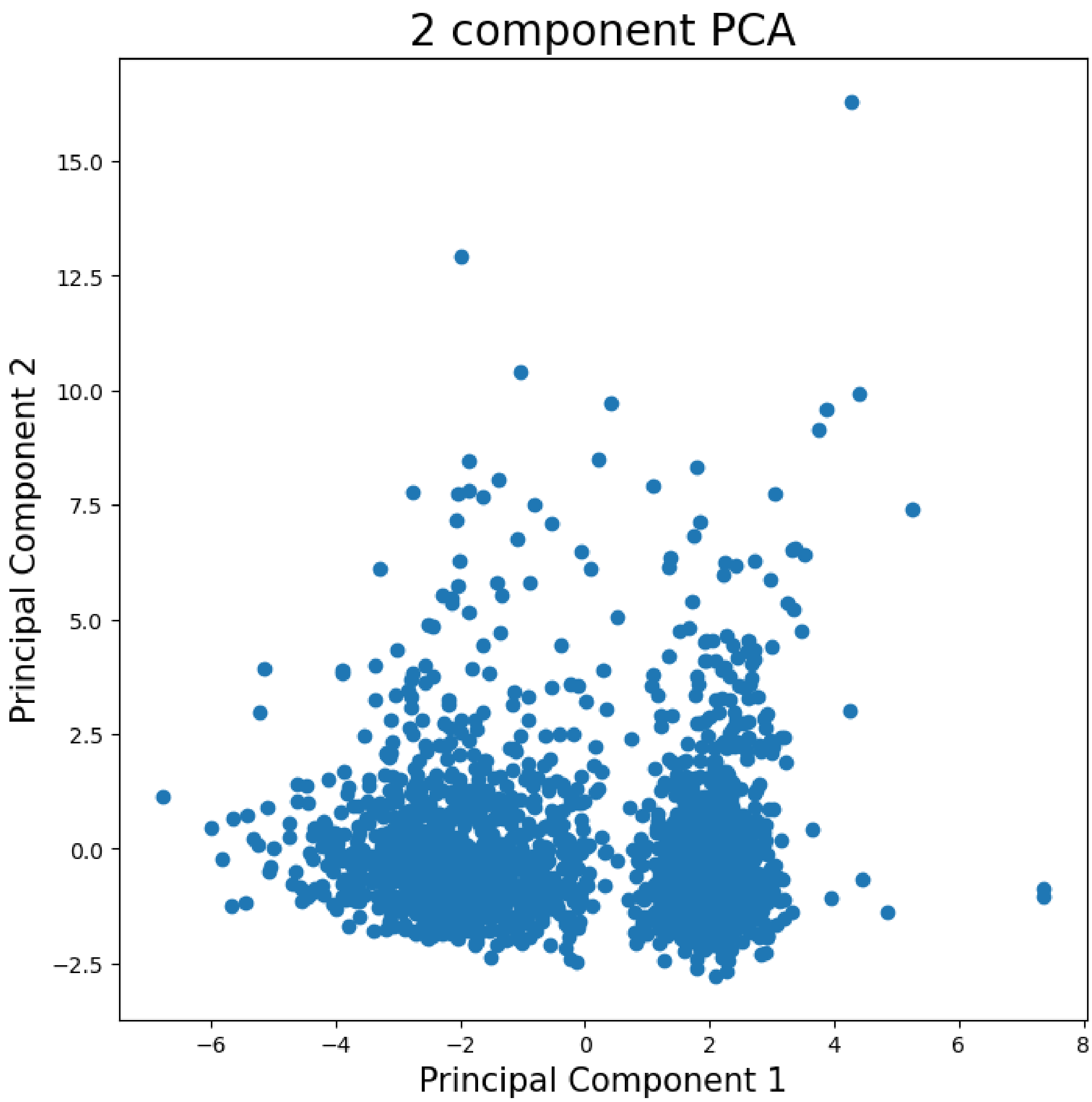
```
['depArr', 'distance', 'distance_long', 'operation_mode', 'angle_sum',
'QDepArr', 'NDepDep', 'NDepArr', 'NArrArr', 'Pressure', 'WindSpeedInMPS',
'isRain', 'aircraft_weight', 'AvgSpdLast5Dep', 'AvgSpdLast5Arr',
'AvgSpdLast10Dep', 'AvgSpdLast10Arr', 'TaxiTime']
['depArr', 'distance', 'distance_long', 'operation_mode', 'angle_sum',
'QDepArr', 'NDepDep', 'NDepArr', 'NArrArr', 'Pressure', 'WindSpeedInMPS',
'isRain', 'aircraft_weight', 'AvgSpdLast5Dep', 'AvgSpdLast5Arr',
'AvgSpdLast10Dep', 'AvgSpdLast10Arr', 'TaxiTime']
```

```
['depArr', 'distance', 'distance_long', 'operation_mode', 'angle_sum',
'QDepArr', 'NDepDep', 'NDepArr', 'NArrArr', 'Pressure', 'WindSpeedInMPS',
'isRain', 'aircraft_weight', 'AvgSpdLast5Dep', 'AvgSpdLast5Arr',
'AvgSpdLast10Dep', 'AvgSpdLast10Arr']
```

[66]: 17

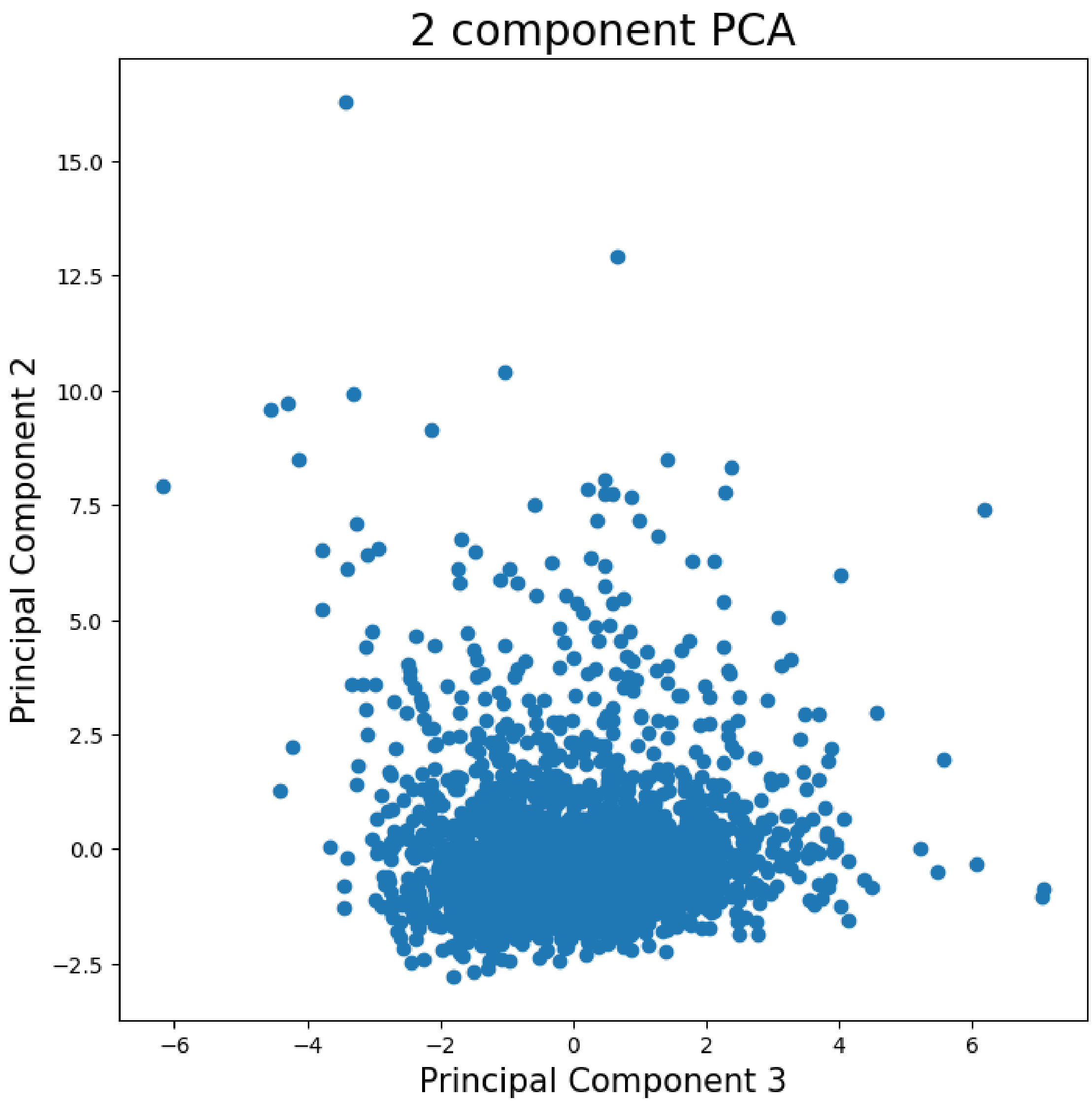
[67]:
```
import matplotlib.pyplot as plt

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[:, 'principal␣
↪component 2'])
```

[67]:
```
<matplotlib.collections.PathCollection at 0x7fcfeffe5750>
```



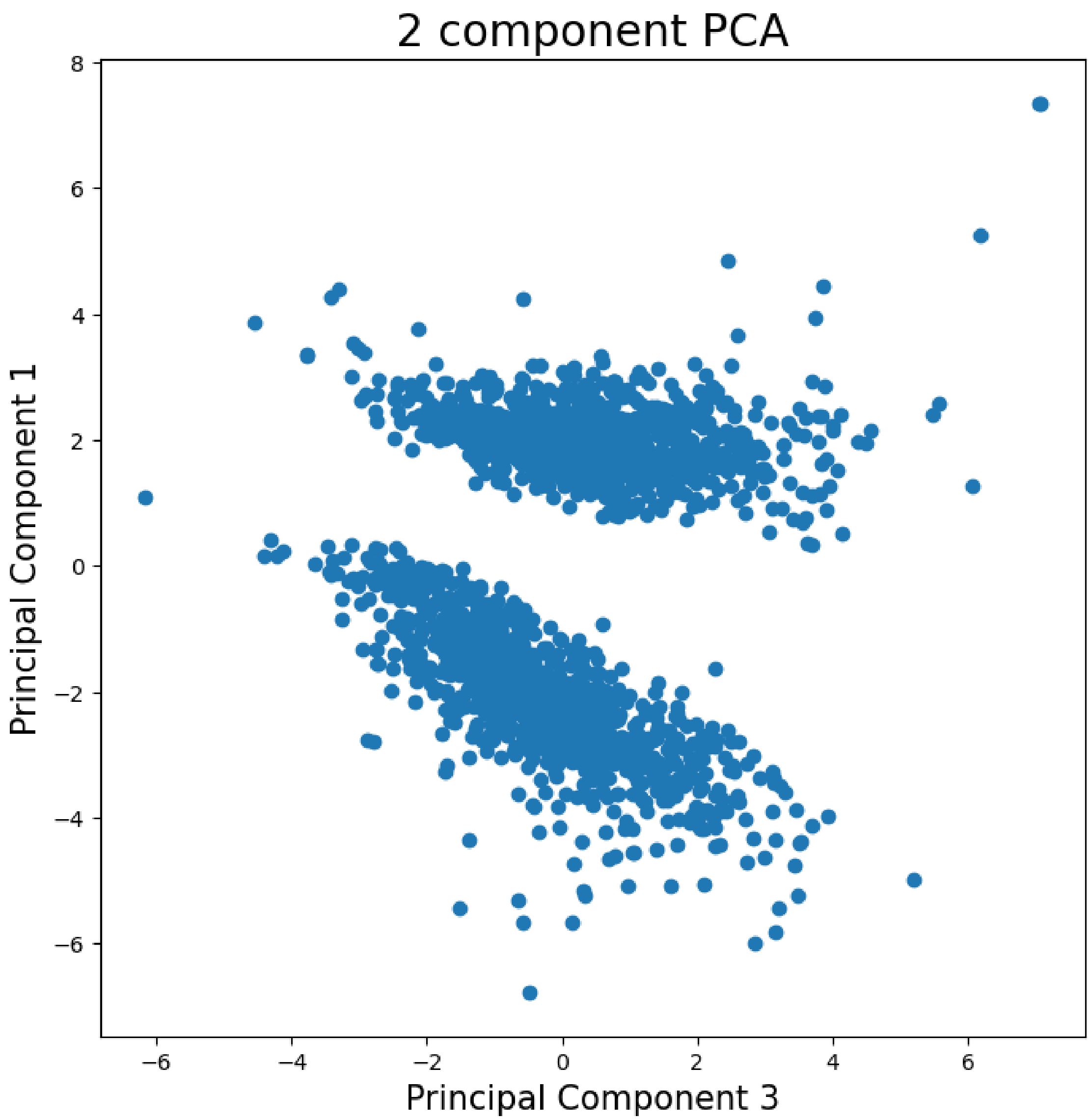
[48]:
```
import matplotlib.pyplot as plt

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 2', fontsize = 15)

ax.set_title('2 component PCA', fontsize = 20)

ax.scatter(finalDf.loc[:, 'principal component 3'], finalDf.loc[:, 'principal␣
↪component 2'])
```

[48]: <matplotlib.collections.PathCollection at 0x7fcffd4be800>



[49]:
```python
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 1', fontsize = 15)

ax.set_title('2 component PCA', fontsize = 20)

ax.scatter(finalDf.loc[:, 'principal component 3'], finalDf.loc[:, 'principal␣
↪component 1'])
```

[49]:
```
<matplotlib.collections.PathCollection at 0x7fcfee5265c0>
```




```
[50]: finalDf.to_csv('data_airport.csv', header=False, index=False)
#define train
trainDf=finalDf.iloc[:int(76*len(finalDf)/100)]
#define test
testDf=finalDf.iloc[int(76*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 component 3  \
```

```
0            -0.533601             0.838938            -1.413943
1             2.812005             0.092036            -0.628742
2             1.659361             0.787785            -0.304296
3             1.573405             1.058287             1.103281
4             1.706248            -2.024809             0.211023
…                    …                    …                    …
1584         -1.497155            -1.770056            -1.034307
1585         -2.779197             0.397704             1.336460
1586         -1.929236            -1.368254            -0.958417
1587          2.134981            -0.199959             2.648513
1588         -1.863990            -0.860810            -1.226913

      principal component 4  TaxiTime
0                  0.741299  4.016667
1                  1.604069  4.015017
2                  0.782495  4.014000
3                 -0.489203  4.011700
4                 -0.762711  4.011450
…                         …         …
1584               0.963861  6.033333
1585              -1.383789  6.016667
1586              -1.515687  6.016667
1587              -2.669789  6.016667
1588               0.249835  6.016667

[1589 rows x 5 columns]
      principal component 1  principal component 2  principal component 3  \
1589               1.622823              -0.784002               1.972534
1590              -2.354210              -0.880873              -0.608626
1591              -1.402329              -0.550196              -0.987337
1592              -1.835577              -0.621371              -0.226616
1593              -1.612064              -0.172162              -1.975470
…                         …                      …                      …
2086               1.996395               0.430197               0.974316
2087               3.180031              -0.654439              -0.320278
2088               2.272796              -2.691960              -1.494102
2089               2.030891               1.455905               0.512584
2090               2.381363               2.644778              -1.079090

      principal component 4  TaxiTime
1589               1.641096  5.970783
1590              -0.611100  5.966667
1591              -0.984404  5.966667
1592               1.021379  5.966667
1593               1.016603  5.966667
…                         …         …
2086              -0.746085  2.471217
2087               0.438766  2.467350
```

```
2088               -0.345529  2.460367
2089                1.361216  2.456500
2090                0.942816  2.452233

[502 rows x 5 columns]
```

[51]:
```
loadings=pca.components_
top_10_indices = np.argsort(explainedVars)[::-1][:10]
for i, index in enumerate (top_10_indices):
    component_name = features[index]
    component_loading = loadings[:,index]
    print(f"{i+1}. {component_name}: {component_loading}")
```

```
1. depArr: [ 0.40718128 -0.0246078   0.14403295  0.01404664  0.01237025
0.00300438
 -0.01416571  0.01672102  0.02916291  0.01507421  0.01253097 -0.0271305
  0.00934355 -0.06190479 -0.16905609  0.25456679  0.07800534  0.2397912
 -0.00646507  0.00595773 -0.01739095  0.02393158  0.0184206   0.05866669
  0.00285435 -0.08255492  0.27432953 -0.74989325  0.         ]
2. distance: [-1.71943873e-01  9.45773071e-02  5.21504166e-01  1.64289954e-01
 -7.25012340e-02  1.33516230e-01 -7.34059627e-02  5.82430873e-02
 -2.37517047e-02  2.61403476e-02 -3.88768974e-02 -2.25942475e-01
  2.84197956e-02 -2.51582871e-02 -8.59788607e-02 -6.21994184e-02
 -7.66095166e-02  7.53286668e-02 -2.06252516e-02  6.41123015e-02
 -3.75287201e-03  8.15053292e-03 -4.25309543e-03  3.84309818e-02
 -1.51149509e-02  7.29207461e-01  1.20546151e-01  4.45344264e-04
 -1.04722092e-17]
3. operation_mode: [-1.88788968e-01  6.16698377e-03  3.79324394e-02
-1.86230978e-02
 -8.14406530e-02  2.88196343e-03 -1.55913712e-02 -1.30969877e-01
  2.08997835e-01 -3.07574949e-01  1.22296298e-01  6.31588370e-01
 -2.16466030e-01 -4.36422702e-01 -2.88849499e-01  2.37210384e-02
 -1.96372123e-01  1.36579981e-01  1.05654079e-02 -1.35306354e-02
  8.95163612e-03  3.05936459e-02  4.01158430e-03  1.85899482e-03
  5.86094699e-03  7.61590969e-02  2.26636897e-02  1.18963518e-02
  2.17294517e-17]
4. angle_sum: [-2.10019804e-01  9.31491654e-02  4.38506755e-01  1.27856694e-01
 -8.50441241e-02  1.06532071e-01 -8.31789154e-02  3.79568279e-02
 -8.23427598e-03  1.03702110e-01  8.47545831e-02 -2.09753646e-01
 -4.30592483e-02  2.27054723e-02 -1.24653369e-01 -2.06086927e-01
 -4.29781934e-01  1.99372047e-01  6.95619311e-03  1.24306732e-02
  4.09382668e-02  4.07140642e-02 -9.45567747e-03 -3.30226665e-02
  8.99744569e-03 -5.99554813e-01 -1.88940231e-02  1.32120108e-02
 -1.81066051e-17]
```

[ ]:

4 Another PCA Example: Breast Cancer

Additional PCA Example – Breast Cancer Dataset This section shows a second PCA application using the well-known breast cancer dataset.

[14]:
```
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()
```

[14]:
```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  …  worst texture  worst perimeter  worst area  \
0                 0.07871  …          17.33           184.60      2019.0
1                 0.05667  …          23.41           158.80      1956.0
2                 0.05999  …          25.53           152.50      1709.0
3                 0.09744  …          26.50            98.87       567.7
4                 0.05883  …          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654
```

```
1          0.1238             0.1866          0.2416                0.1860
2          0.1444             0.4245          0.4504                0.2430
3          0.2098             0.8663          0.6869                0.2575
4          0.1374             0.2050          0.4000                0.1625

   worst symmetry  worst fractal dimension  label
0          0.4601                  0.11890    0.0
1          0.2750                  0.08902    0.0
2          0.3613                  0.08758    0.0
3          0.6638                  0.17300    0.0
4          0.2364                  0.07678    0.0

[5 rows x 31 columns]
```

[15]:
```
breast_dataset['label'].replace(0, 'Benign',inplace=True)
breast_dataset['label'].replace(1, 'Malignant',inplace=True)

breast_dataset.tail()
```

[15]:
```
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
564        21.56         22.39          142.00     1479.0          0.11100
565        20.13         28.25          131.20     1261.0          0.09780
566        16.60         28.08          108.30      858.1          0.08455
567        20.60         29.33          140.10     1265.0          0.11780
568         7.76         24.54           47.92      181.0          0.05263

     mean compactness  mean concavity  mean concave points  mean symmetry  \
564           0.11590         0.24390              0.13890         0.1726
565           0.10340         0.14400              0.09791         0.1752
566           0.10230         0.09251              0.05302         0.1590
567           0.27700         0.35140              0.15200         0.2397
568           0.04362         0.00000              0.00000         0.1587

     mean fractal dimension  …  worst texture  worst perimeter  worst area  \
564                 0.05623  …          26.40           166.10      2027.0
565                 0.05533  …          38.25           155.00      1731.0
566                 0.05648  …          34.12           126.70      1124.0
567                 0.07016  …          39.42           184.60      1821.0
568                 0.05884  …          30.37            59.16       268.6

     worst smoothness  worst compactness  worst concavity  \
564           0.14100            0.21130           0.4107
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
568           0.08996            0.06444           0.0000
```

```
     worst concave points  worst symmetry  worst fractal dimension      label
564                0.2216          0.2060                  0.07115     Benign
565                0.1628          0.2572                  0.06637     Benign
566                0.1418          0.2218                  0.07820     Benign
567                0.2650          0.4087                  0.12400     Benign
568                0.0000          0.2871                  0.07039  Malignant

[5 rows x 31 columns]
```

[16]:

```python
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()
```

[16]:

```
     feature0  feature1  feature2  feature3  feature4  feature5  feature6  \
564  2.110995  0.721473  2.060786  2.343856  1.041842  0.219060  1.947285
565  1.704854  2.085134  1.615931  1.723842  0.102458 -0.017833  0.693043
566  0.702284  2.045574  0.672676  0.577953 -0.840484 -0.038680  0.046588
567  1.838341  2.336457  1.982524  1.735218  1.525767  3.272144  3.296944
568 -1.808401  1.221792 -1.814389 -1.347789 -3.112085 -1.150752 -1.114873

     feature7  feature8  feature9  …  feature20  feature21  feature22  \
564  2.320965 -0.312589 -0.931027  …   1.901185   0.117700   1.752563
565  1.263669 -0.217664 -1.058611  …   1.536720   2.047399   1.421940
566  0.105777 -0.809117 -0.895587  …   0.561361   1.374854   0.579001
567  2.658866  2.137194  1.043695  …   1.961239   2.237926   2.303601
568 -1.261820 -0.820070 -0.561032  …  -1.410893   0.764190  -1.432735

     feature23  feature24  feature25  feature26  feature27  feature28  \
564   2.015301   0.378365  -0.273318   0.664512   1.629151  -1.360158
565   1.494959  -0.691230  -0.394820   0.236573   0.733827  -0.531855
566   0.427906  -0.809587   0.350735   0.326767   0.414069  -1.104549
567   1.653171   1.430427   3.904848   3.197605   2.289985   1.919083
568  -1.075813  -1.859019  -1.207552  -1.305831  -1.745063  -0.048138

     feature29
564  -0.709091
565  -0.973978
566  -0.318409
567   2.219635
568  -0.751207

[5 rows x 30 columns]
```

[17]:
```python
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_breast,
                                   columns = ['principal component 1',␣
 ↪'principal component 2'])

principal_breast_Df.tail()
```

[17]:
```
     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
```

[18]:
```python
print('Explained variation per principal component: {}'.format(pca_breast.
 ↪explained_variance_ratio_))
```

```
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.

[19]:
```python
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",fontsize=20)
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 component 1'],
                principal_breast_Df.loc[indicesToKeep, 'principal component␣
 ↪2'], c = color, s = 50)
```

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

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