

## 3.0.PCA

May 15, 2023

```
[58]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import time
from subprocess import check_output
from scipy import stats
plt.style.use("ggplot")
import warnings
warnings.filterwarnings("ignore")
```

```
[59]: data=pd.read_csv('wdbc.data',header=None)
```

```
data.head()
```

```
[60]: data.head()
```

```
[60]:
```

	0	1	2	3	4	5	6	7	8	
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	\
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

	9	...	22	23	24	25	26	27	28	29	
0	0.14710	...	25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	\
1	0.07017	...	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	
2	0.12790	...	23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	
3	0.10520	...	14.91	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	
4	0.10430	...	22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	

	30	31
0	0.4601	0.11890
1	0.2750	0.08902
2	0.3613	0.08758
3	0.6638	0.17300
4	0.2364	0.07678

[5 rows x 32 columns]

```
[61]: headers=['id','diagnosis','mean_radius','mean_texture','mean_perimeter','mean_area','mean_smoothness',
↳points','mean_symmetry','mean_fractal_dimension','se','se_points','se_symmetry','se_fractal_dimension',
↳dimension','worst_radius','worst_texture','worst_perimeter','worst_area','worst_smoothness',
↳points','worst_symmetry','worst_fractal_dimension']
```

```
[62]: data.to_csv('labeledData.csv',header=headers,index=False)
```

```
[63]: data=pd.read_csv('labeledData.csv')
data.head()
```

```
[63]:
```

	id	diagnosis	mean_radius	mean_texture	mean_perimeter	mean_area	
0	842302	M	17.99	10.38	122.80	1001.0	\
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

	mean_smoothness	mean_compactness	mean_concavity	mean_concave points	
0	0.11840	0.27760	0.3001	0.14710	\
1	0.08474	0.07864	0.0869	0.07017	
2	0.10960	0.15990	0.1974	0.12790	
3	0.14250	0.28390	0.2414	0.10520	
4	0.10030	0.13280	0.1980	0.10430	

...	worst_radius	worst_texture	worst_perimeter	worst_area	
0	25.38	17.33	184.60	2019.0	\
1	24.99	23.41	158.80	1956.0	
2	23.57	25.53	152.50	1709.0	
3	14.91	26.50	98.87	567.7	
4	22.54	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
0	0.4601	0.11890
1	0.2750	0.08902
2	0.3613	0.08758
3	0.6638	0.17300

4                    0.2364                    0.07678

[5 rows x 32 columns]

```
[64]: def diag(z):
      if z=='M':
          return 1
      else:
          return 0
      z=data['diagnosis'].apply(diag)
      data.diagnosis=z
```

```
[65]: df=pd.DataFrame(data)
      df=df.drop('id',axis=1)
      df
```

```
[65]:
```

	diagnosis	mean_radius	mean_texture	mean_perimeter	mean_area	
0	1	17.99	10.38	122.80	1001.0	\
1	1	20.57	17.77	132.90	1326.0	
2	1	19.69	21.25	130.00	1203.0	
3	1	11.42	20.38	77.58	386.1	
4	1	20.29	14.34	135.10	1297.0	
..	...	...	...	...	...	
564	1	21.56	22.39	142.00	1479.0	
565	1	20.13	28.25	131.20	1261.0	
566	1	16.60	28.08	108.30	858.1	
567	1	20.60	29.33	140.10	1265.0	
568	0	7.76	24.54	47.92	181.0	

	mean_smoothness	mean_compactness	mean_concavity	mean_concave	points	
0	0.11840	0.27760	0.30010		0.14710	\
1	0.08474	0.07864	0.08690		0.07017	
2	0.10960	0.15990	0.19740		0.12790	
3	0.14250	0.28390	0.24140		0.10520	
4	0.10030	0.13280	0.19800		0.10430	
..	...	...	...		...	
564	0.11100	0.11590	0.24390		0.13890	
565	0.09780	0.10340	0.14400		0.09791	
566	0.08455	0.10230	0.09251		0.05302	
567	0.11780	0.27700	0.35140		0.15200	
568	0.05263	0.04362	0.00000		0.00000	

	mean_symmetry	...	worst_radius	worst_texture	worst_perimeter	
0	0.2419	...	25.380	17.33	184.60	\
1	0.1812	...	24.990	23.41	158.80	
2	0.2069	...	23.570	25.53	152.50	
3	0.2597	...	14.910	26.50	98.87	

4	0.1809	...	22.540	16.67	152.20
..	...	...	...	...	...
564	0.1726	...	25.450	26.40	166.10
565	0.1752	...	23.690	38.25	155.00
566	0.1590	...	18.980	34.12	126.70
567	0.2397	...	25.740	39.42	184.60
568	0.1587	...	9.456	30.37	59.16

	worst_area	worst_smoothness	worst_compactness	worst_concavity	
0	2019.0	0.16220	0.66560	0.7119	\
1	1956.0	0.12380	0.18660	0.2416	
2	1709.0	0.14440	0.42450	0.4504	
3	567.7	0.20980	0.86630	0.6869	
4	1575.0	0.13740	0.20500	0.4000	
..	...	...	...	...	
564	2027.0	0.14100	0.21130	0.4107	
565	1731.0	0.11660	0.19220	0.3215	
566	1124.0	0.11390	0.30940	0.3403	
567	1821.0	0.16500	0.86810	0.9387	
568	268.6	0.08996	0.06444	0.0000	

	worst_concave points	worst_symmetry	worst_fractal dimension
0	0.2654	0.4601	0.11890
1	0.1860	0.2750	0.08902
2	0.2430	0.3613	0.08758
3	0.2575	0.6638	0.17300
4	0.1625	0.2364	0.07678
..	...	...	...
564	0.2216	0.2060	0.07115
565	0.1628	0.2572	0.06637
566	0.1418	0.2218	0.07820
567	0.2650	0.4087	0.12400
568	0.0000	0.2871	0.07039

[569 rows x 31 columns]

```
[66]: x=df.drop('diagnosis',axis=1)
      x.shape
```

```
[66]: (569, 30)
```

```
[67]: y=df.iloc[:,0:1]
      y.shape
```

```
[67]: (569, 1)
```

```
[68]: pip install scikit-learn
```

Requirement already satisfied: scikit-learn in  
 c:\users\sheel\appdata\local\programs\python\python311\lib\site-packages (1.2.2)  
 Requirement already satisfied: numpy>=1.17.3 in  
 c:\users\sheel\appdata\local\programs\python\python311\lib\site-packages (from  
 scikit-learn) (1.24.3)  
 Requirement already satisfied: scipy>=1.3.2 in  
 c:\users\sheel\appdata\local\programs\python\python311\lib\site-packages (from  
 scikit-learn) (1.10.1)  
 Requirement already satisfied: joblib>=1.1.1 in  
 c:\users\sheel\appdata\local\programs\python\python311\lib\site-packages (from  
 scikit-learn) (1.2.0)  
 Requirement already satisfied: threadpoolctl>=2.0.0 in  
 c:\users\sheel\appdata\local\programs\python\python311\lib\site-packages (from  
 scikit-learn) (3.1.0)  
 Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip available: 22.3.1 -> 23.1.2

[notice] To update, run: python.exe -m pip install --upgrade pip

```
[69]: from sklearn.preprocessing import StandardScaler
x = df.iloc[:, :-1]
y = df.iloc[:, -1]
# performing standardization
sc = StandardScaler()
x_scaled = sc.fit_transform(x)
```

```
[70]: from sklearn.decomposition import PCA
components=None
pca=PCA(n_components=components)
pca.fit(x_scaled)
```

[70]: PCA()

```
[71]: print('Cumulative Variances Percentage:')
print(pca.explained_variance_ratio_.cumsum()*100)
```

Cumulative Variances Percentage:

```
[ 45.70482024  63.43807513  72.38144906  79.00927945  84.47359555
  88.47267763  90.44353313  92.04384397  93.39708966  94.56672126
  95.5371204   96.48654887  97.35602624  98.06158498  98.53894853
  98.82464805  99.03516744  99.21059817  99.38127455  99.54166534
  99.64391818  99.74210062  99.82305224  99.8830052   99.94149718
  99.96835313  99.99168096  99.99706656  99.99955739 100.         ]
```

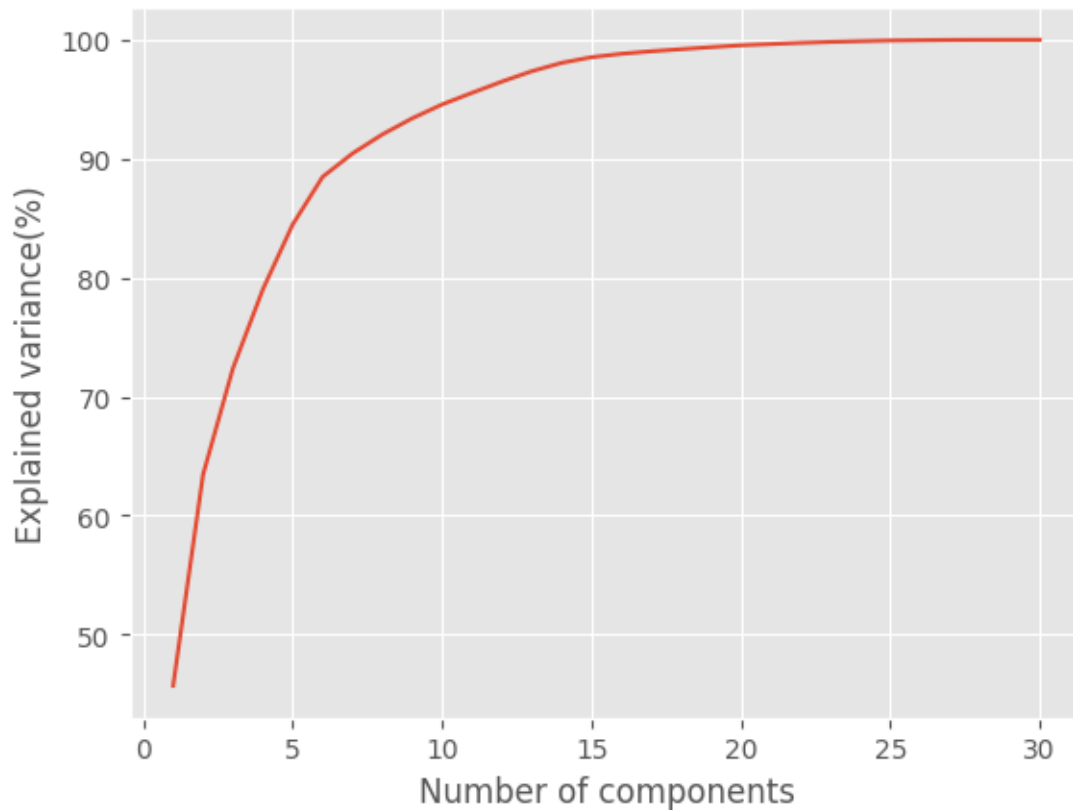
```
[72]: components=len(pca.explained_variance_ratio_)
      if components is None else components
plt.plot(range(1, components+1),
```

```

        np.cumsum(pca.explained_variance_ratio_*100))
plt.xlabel('Number of components')
plt.ylabel('Explained variance(%)')

```

[72]: Text(0, 0.5, 'Explained variance(%)')



```

[73]: from sklearn.decomposition import PCA
pca=PCA(n_components=0.85)
pca.fit(x_scaled)
print('Cumulative Variances (Percentage):')
print(np.cumsum(pca.explained_variance_ratio_*100))
components=len(pca.explained_variance_ratio_)
print(f'Number of components:{components}')
plt.plot(range(1,components+1),
np.cumsum(pca.explained_variance_ratio_*100))
plt.xlabel('Number of components')
plt.ylabel('Explained Variance (%)')

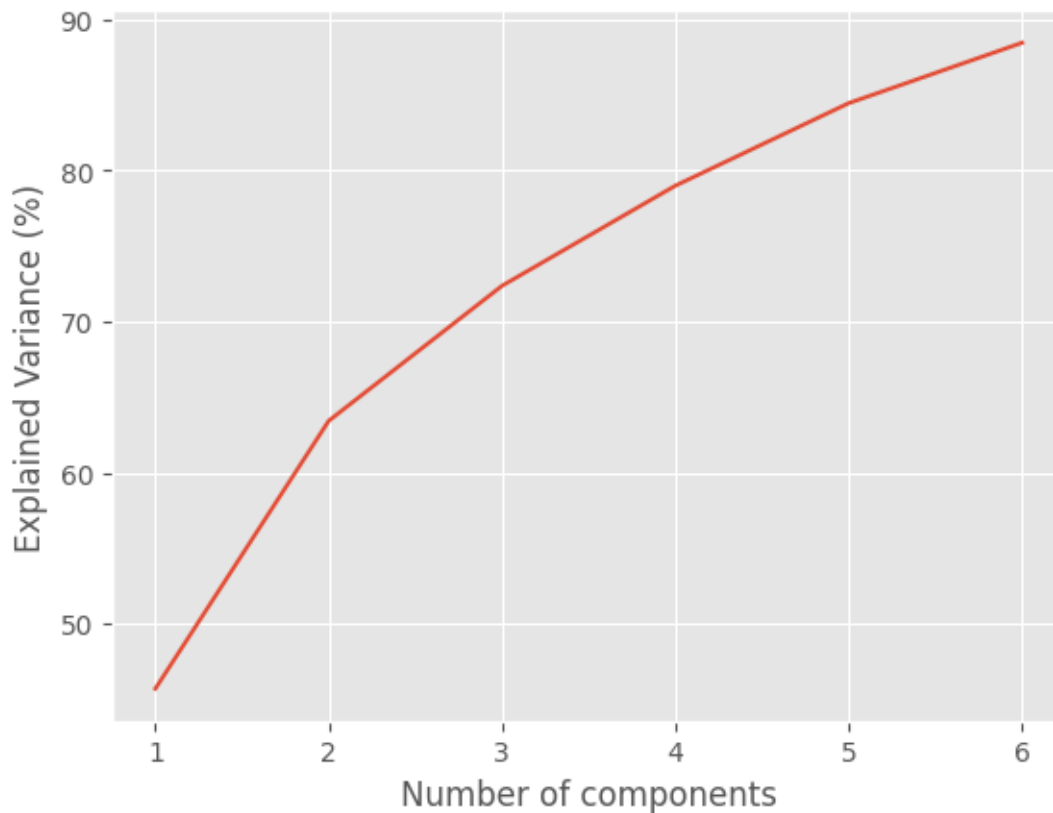
```

Cumulative Variances (Percentage):

[45.70482024 63.43807513 72.38144906 79.00927945 84.47359555 88.47267763]

Number of components:6

```
[73]: Text(0, 0.5, 'Explained Variance (%)')
```



```
[74]: pca_components=abs(pca.components_)
print(pca_components)
```

```
[[0.21946075 0.22327147 0.1053675  0.23103393 0.22492103 0.13445769
 0.22899318 0.25161962 0.25792787 0.13031253 0.04899745 0.20622073
 0.01582104 0.21057403 0.20387997 0.00910145 0.1582188  0.1434352
 0.17645803 0.03719747 0.08953781 0.23171618 0.10566991 0.23935141
 0.22817589 0.12069915 0.20012565 0.22073753 0.24704713 0.11695725]
[0.07527256 0.22054497 0.05604209 0.20063176 0.21506452 0.20205704
 0.17192616 0.08570235 0.01229753 0.21196099 0.37647526 0.07087837
 0.11858407 0.05388621 0.12318004 0.2364953  0.26339904 0.23030467
 0.17047432 0.22040844 0.30600865 0.21009518 0.05035212 0.18876594
 0.20748237 0.16936642 0.14271038 0.10471416 0.00081362 0.13905316]
[0.10381349 0.02078243 0.08656462 0.01724522 0.05956234 0.15956682
 0.10068951 0.01046439 0.03298289 0.09744592 0.06879573 0.2831986
 0.36669189 0.28069702 0.24095374 0.26840399 0.13395417 0.15592669
 0.20074335 0.24173635 0.19370193 0.01903268 0.02431895 0.02253289
 0.01916973 0.3058468  0.25009862 0.18447942 0.17829786 0.3172018 ]
[0.10782484 0.05598946 0.5964038  0.05604853 0.0689083  0.12181867
```

```

0.0290586 0.02469142 0.06382697 0.03194798 0.03636076 0.10476435
0.36603356 0.09828958 0.12069604 0.02262799 0.01017672 0.02116795
0.0825686 0.01874927 0.03337408 0.02696833 0.63235721 0.02579059
0.03916518 0.01945712 0.08802775 0.06645568 0.00523576 0.07097062]
[0.068848 0.03552887 0.00068558 0.03627585 0.0024271 0.35814432
0.03106979 0.09944786 0.03717768 0.280422 0.04035745 0.18967918
0.18511124 0.15281022 0.16114979 0.25940583 0.28622685 0.36003656
0.18895704 0.2570678 0.24837444 0.00391082 0.03207867 0.0099555
0.03237304 0.30066745 0.15878282 0.22212248 0.06679247 0.19599699]
[0.01708368 0.01969708 0.03586663 0.01780585 0.00419642 0.28976943
0.02281195 0.02119414 0.05782111 0.3547249 0.1175639 0.00151703
0.01980376 0.02711379 0.01468767 0.33965698 0.05611072 0.0326232
0.05110283 0.50061342 0.05492689 0.00414661 0.05566962 0.01189957
0.01549578 0.37729742 0.0355155 0.00833758 0.04581019 0.49327568]]

```

```

[78]: print('Top 4 most important features in each component')
print('=====')
for row in range(pca_components.shape[0]):
    # get the indices of the top 4 values in each row
    temp = np.argpartition(-(pca_components[row]), 4)

    # sort the indices in descending order
    indices = temp[np.argsort((-pca_components[row])[temp]))[:4]

    # print the top 4 feature names
    df2=df.drop('diagnosis',axis=1)
    print(f'Component {row}: {df2.columns[indices].to_list()}')

```

```

Top 4 most important features in each component
=====
Component 0: ['mean_symmetry', 'mean_concave points', 'worst_symmetry',
'worst_area']
Component 1: ['SE_radius', 'worst_radius', 'SE_concavity', 'SE_compactness']
Component 2: ['SE_perimeter', 'worst_fractal dimension', 'worst_compactness',
'SE_texture']
Component 3: ['worst_perimeter', 'mean_perimeter', 'SE_perimeter',
'mean_compactness']
Component 4: ['SE_concave points', 'mean_compactness', 'worst_compactness',
'SE_concavity']
Component 5: ['SE_fractal dimension', 'worst_fractal dimension',
'worst_compactness', 'mean_fractal dimension']

```

```

[79]: x_pca=pca.transform(x_scaled)
print(x_pca.shape)
print(x_pca)

```

```

(569, 6)
[[ 9.00574762  2.13297193 -1.3058032  3.58251042 -1.3601952  1.5555042 ]

```



```
[ 2.72416163 -3.86854352 -0.06524162  1.23678875  0.36458132  0.11451479]
[ 5.9312751  -0.77891015 -0.63084536  0.87193931 -0.1362758  0.49621953]
...
[ 1.62139901 -1.8598167   0.76477174 -1.98279168  1.85452804 -0.6155452 ]
[10.15343163  1.73263141 -1.79963412 -2.27017331  0.41465922  0.52109452]
[-5.43024181 -0.82933047  1.65896772 -2.27087421 -0.17550628  1.72452175]]
```