3.1.PCA-Log-Reg

May 15, 2023

[193]:
```
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")
```

[194]:
```
data=pd.read_csv('wdbc.data',header=None)
```

data.head()

[195]:
```
data.head()
```

[195]:

	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]
```

[196]:
```
headers=['id','diagnosis','mean_radius','mean_texture','mean_perimeter','mean_area','mean_smoo
 ↪points','mean_symmetry','mean_fractal␣
 ↪dimension','SE_radius','SE_texture','SE_perimeter','SE_area','SE_smoothness','SE_compactnes
 ↪points','SE_symmetry','SE_fractal␣
 ↪dimension','worst_radius','worst_texture','worst_perimeter','worst_area','worst_smoothness'
 ↪points','worst_symmetry','worst_fractal dimension']
```

[197]:
```
data.to_csv('labeledData.csv',header=headers,index=False)
```

[198]:
```
data=pd.read_csv('labeledData.csv')
data.head()
```

[198]:

	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]
```

[199]:
```
def diag(z):
    if z=='M':
        return 1
    else:
        return 0
z=data['diagnosis'].apply(diag)
data.diagnosis=z
```

[200]:
```
df=pd.DataFrame(data)
df=df.drop('id',axis=1)
df
```

[200]:
```
     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]
```

[201]:
```
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
```

[202]:
```
from sklearn.preprocessing import StandardScaler
x = df.iloc[:,1:31]
y = df.iloc[:,0:1]
# performing standardization
sc = StandardScaler()
x_scaled = sc.fit_transform(x)
```

[203]:
```
from sklearn.decomposition import PCA
components=None
pca=PCA(n_components=components)
pca.fit(x_scaled)
```

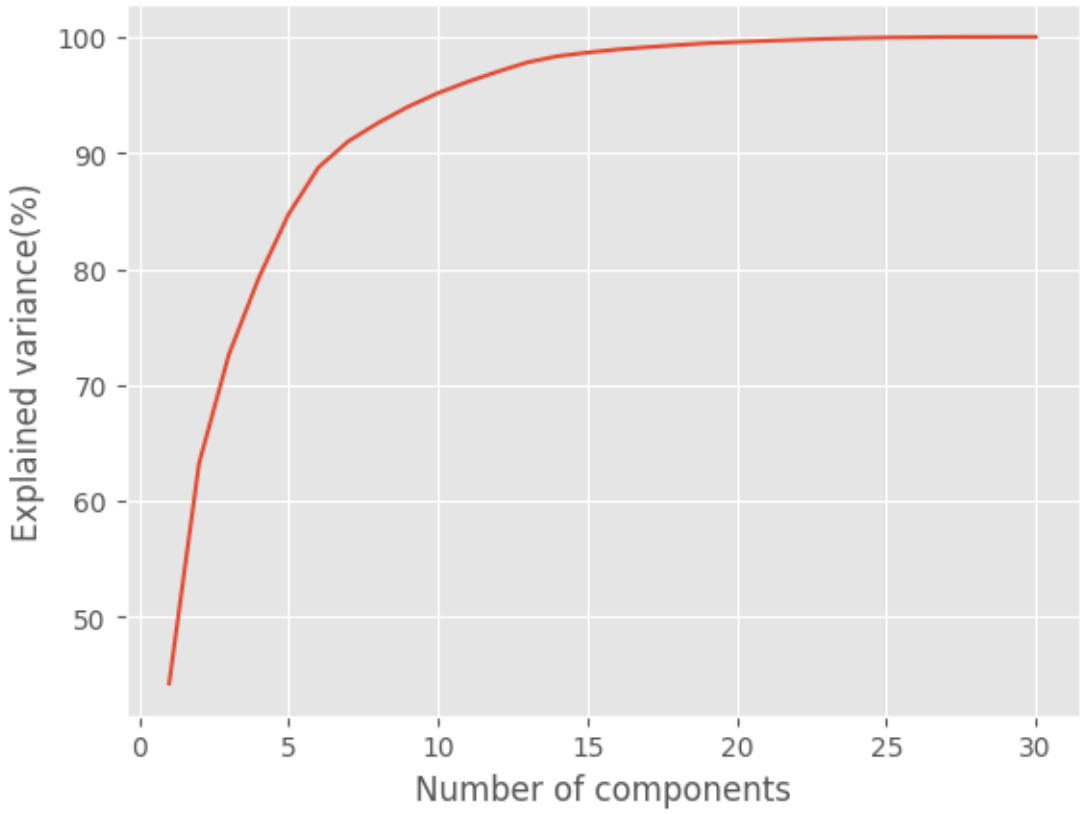
[203]:
```
PCA()
```

[204]:
```
print('Cumulative Variances Percentage:')
print(pca.explained_variance_ratio_.cumsum()*100)
```

```
Cumulative Variances Percentage:
[ 44.27202561  63.24320765  72.63637091  79.23850582  84.73427432
  88.75879636  91.00953007  92.59825387  93.98790324  95.15688143
  96.13660042  97.00713832  97.81166331  98.33502905  98.64881227
  98.91502161  99.1130184   99.28841435  99.45333965  99.55720433
  99.65711397  99.74857865  99.82971477  99.88989813  99.94150237
  99.96876117  99.99176271  99.99706051  99.99955652 100.        ]
```

[205]:
```
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(%)')
```

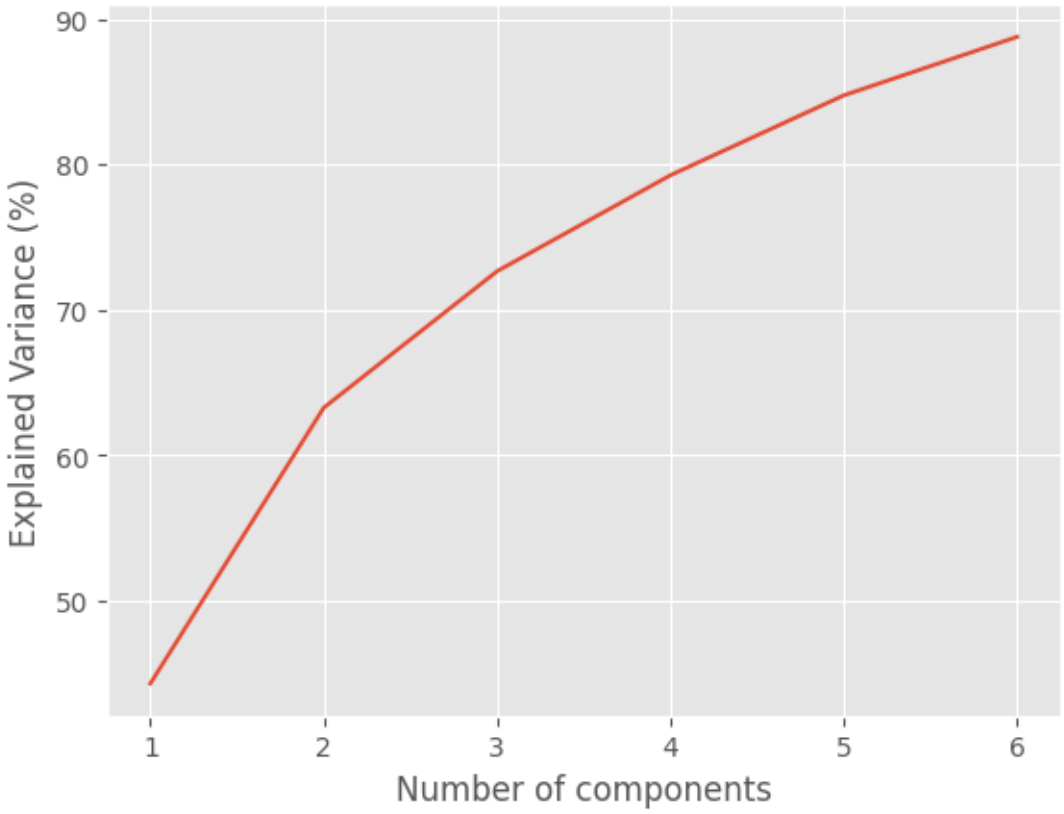
[205]:
```
Text(0, 0.5, 'Explained variance(%)')
```



[206]:
```
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):
[44.27202561 63.24320765 72.63637091 79.23850582 84.73427432 88.75879636]
Number of components:6
```

[206]:
```
Text(0, 0.5, 'Explained Variance (%)')
```



[207]:
```
pca_components=abs(pca.components_)
print(pca_components)
```

```
[[2.18902444e-01 1.03724578e-01 2.27537293e-01 2.20994985e-01
  1.42589694e-01 2.39285354e-01 2.58400481e-01 2.60853758e-01
  1.38166959e-01 6.43633464e-02 2.05978776e-01 1.74280281e-02
  2.11325916e-01 2.02869635e-01 1.45314521e-02 1.70393451e-01
  1.53589790e-01 1.83417397e-01 4.24984216e-02 1.02568322e-01
  2.27996634e-01 1.04469325e-01 2.36639681e-01 2.24870533e-01
  1.27952561e-01 2.10095880e-01 2.28767533e-01 2.50885971e-01
  1.22904556e-01 1.31783943e-01]
 [2.33857132e-01 5.97060883e-02 2.15181361e-01 2.31076711e-01
  1.86113023e-01 1.51891610e-01 6.01653628e-02 3.47675005e-02
  1.90348770e-01 3.66575471e-01 1.05552152e-01 8.99796818e-02
  8.94572342e-02 1.52292628e-01 2.04430453e-01 2.32715896e-01
  1.97207283e-01 1.30321560e-01 1.83848000e-01 2.80092027e-01
  2.19866379e-01 4.54672983e-02 1.99878428e-01 2.19351858e-01
  1.72304352e-01 1.43593173e-01 9.79641143e-02 8.25723507e-03
  1.41883349e-01 2.75339469e-01]
 [8.53124284e-03 6.45499033e-02 9.31421972e-03 2.86995259e-02
  1.04291904e-01 7.40915709e-02 2.73383798e-03 2.55635406e-02
```

```
 4.02399363e-02 2.25740897e-02 2.68481387e-01 3.74633665e-01
 2.66645367e-01 2.16006528e-01 3.08838979e-01 1.54779718e-01
 1.76463743e-01 2.24657567e-01 2.88584292e-01 2.11503764e-01
 4.75069900e-02 4.22978228e-02 4.85465083e-02 1.19023182e-02
 2.59797613e-01 2.36075625e-01 1.73057335e-01 1.70344076e-01
 2.71312642e-01 2.32791313e-01]
[4.14089623e-02 6.03050001e-01 4.19830991e-02 5.34337955e-02
 1.59382765e-01 3.17945811e-02 1.91227535e-02 6.53359443e-02
 6.71249840e-02 4.85867649e-02 9.79412418e-02 3.59855528e-01
 8.89924146e-02 1.08205039e-01 4.46641797e-02 2.74693632e-02
 1.31687997e-03 7.40673350e-02 4.40733510e-02 1.53047496e-02
 1.54172396e-02 6.32807885e-01 1.38027944e-02 2.58947492e-02
 1.76522161e-02 9.13284153e-02 7.39511797e-02 6.00699571e-03
 3.62506947e-02 7.70534703e-02]
[3.77863538e-02 4.94688505e-02 3.73746632e-02 1.03312514e-02
 3.65088528e-01 1.17039713e-02 8.63754118e-02 4.38610252e-02
 3.05941428e-01 4.44243602e-02 1.54456496e-01 1.91650506e-01
 1.20990220e-01 1.27574432e-01 2.32065676e-01 2.79968156e-01
 3.53982091e-01 1.95548089e-01 2.52868765e-01 2.63297438e-01
 4.40659209e-03 9.28834001e-02 7.45415100e-03 2.73909030e-02
 3.24435445e-01 1.21804107e-01 1.88518727e-01 4.33320687e-02
 2.44558663e-01 9.44233510e-02]
[1.87407904e-02 3.21788366e-02 1.73084449e-02 1.88774796e-03
 2.86374497e-01 1.41309489e-02 9.34418089e-03 5.20499505e-02
 3.56458461e-01 1.19430668e-01 2.56032561e-02 2.87473145e-02
 1.81071500e-03 4.28639079e-02 3.42917393e-01 6.91975186e-02
 5.63432386e-02 3.12244482e-02 4.90245643e-01 5.31952674e-02
 2.90684919e-04 5.00080613e-02 8.50098715e-03 2.51643821e-02
 3.69255370e-01 4.77057929e-02 2.83792555e-02 3.08734498e-02
 4.98926784e-01 8.02235245e-02]]
```

[208]:
```python
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_concave points', 'mean_concavity', 'worst_concave points',
'mean_compactness']
```

```
Component 1: ['mean_fractal dimension', 'SE_fractal dimension', 'worst_fractal
dimension', 'mean_radius']
Component 2: ['SE_texture', 'SE_smoothness', 'SE_symmetry', 'worst_symmetry']
Component 3: ['worst_texture', 'mean_texture', 'SE_texture', 'mean_smoothness']
Component 4: ['mean_smoothness', 'SE_concavity', 'worst_smoothness',
'mean_symmetry']
Component 5: ['worst_symmetry', 'SE_symmetry', 'worst_smoothness',
'mean_symmetry']
```

[209]:
```
x_pca=pca.transform(x_scaled)
print(x_pca.shape)
print(x_pca)
```

```
(569, 6)
[[ 9.19283683  1.94858307 -1.12316616  3.6337309  -1.19511012  1.41142445]
 [ 2.3878018  -3.76817174 -0.52929269  1.11826386  0.62177498  0.02865635]
 [ 5.73389628 -1.0751738  -0.55174759  0.91208267 -0.1770859   0.54145215]
 …
 [ 1.25617928 -1.90229671  0.56273053 -2.08922702  1.80999133 -0.53444719]
 [10.37479406  1.67201011 -1.87702933 -2.35603113 -0.03374193  0.56793647]
 [-5.4752433  -0.67063679  1.49044308 -2.29915714 -0.18470331  1.61783736]]
```

[210]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
random_state=42
x_train,x_test,y_train,y_test=\
train_test_split(x_scaled,y,test_size=0.
 ↪3,shuffle=True,random_state=random_state)
log_reg=LogisticRegression()
log_reg.fit(x_train,y_train)
log_reg.score(x_test,y_test)
```

[210]: 0.9824561403508771

[211]:
```
random_state=42
x_pca_train,x_pca_test,y_pca_train,y_pca_test=\
train_test_split(x_pca,y,test_size=0.3,shuffle=True,random_state=random_state)
log_reg_pca=LogisticRegression()
log_reg_pca.fit(x_pca_train,y_pca_train)
log_reg_pca.score(x_pca_test,y_pca_test)
```

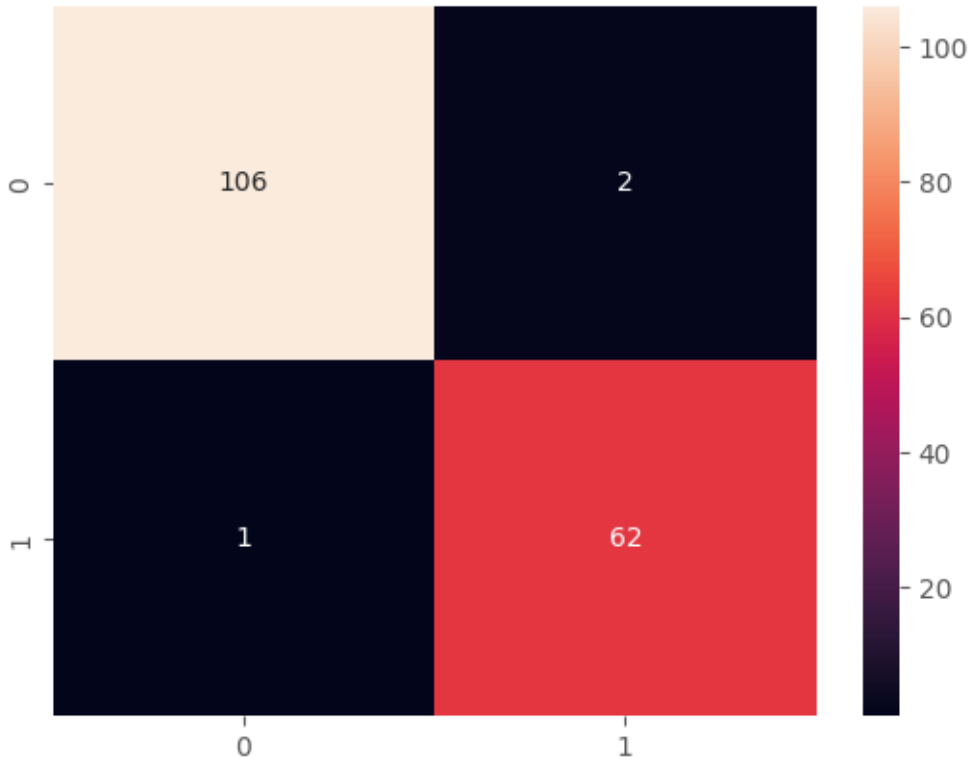
[211]: 0.9883040935672515

[212]:
```
from sklearn.metrics import f1_score,confusion_matrix
from sklearn.metrics import accuracy_score
acc=accuracy_score(y_test,log_reg.predict(x_test))
print('Accuracy is:',acc)
```

```
cm=confusion_matrix(y_test,log_reg.predict(x_test))
sns.heatmap(cm,annot=True,fmt='d')
```

```
Accuracy is: 0.9824561403508771
```

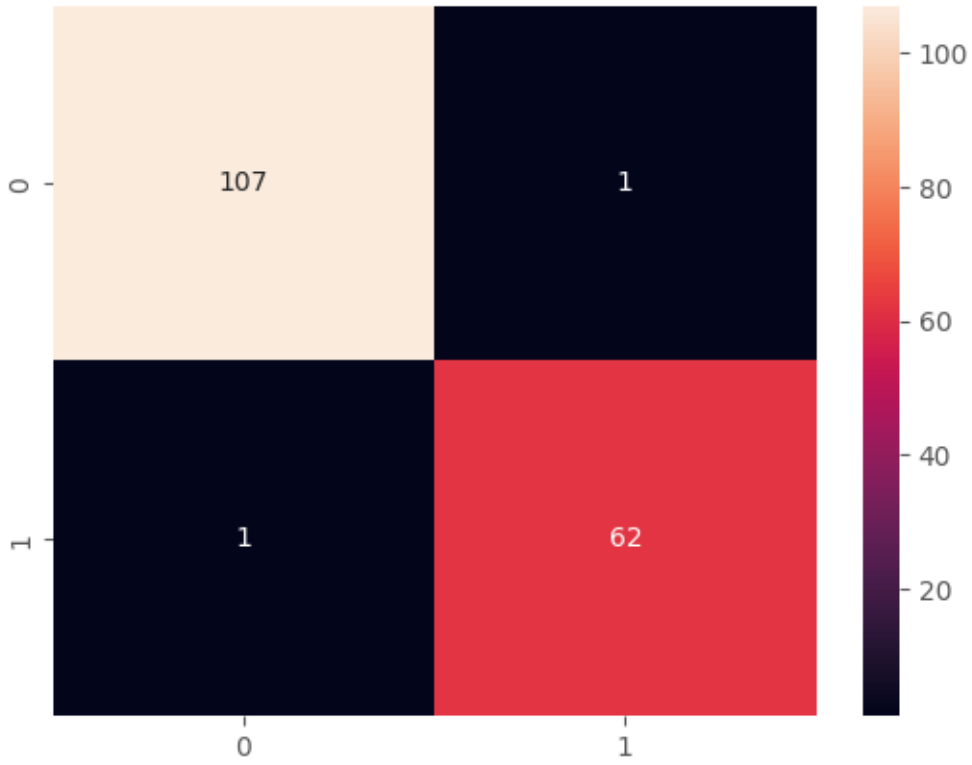
[212]: <Axes: >



[213]:
```
acc_pca=accuracy_score(y_pca_test,log_reg_pca.predict(x_pca_test))
print('Accuracy is:',acc_pca)
cm=confusion_matrix(y_pca_test,log_reg_pca.predict(x_pca_test))
sns.heatmap(cm,annot=True,fmt='d')
```

```
Accuracy is: 0.9883040935672515
```

[213]: <Axes: >


107
1
1
62
0
1
0
1
100
80
60
40
20