5.3.KNN-Analyze

May 15, 2023

[1]:
```
# KNN Classification
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
%matplotlib inline
from sklearn.model_selection import train_test_split
from scipy.stats import zscore
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
data=pd.read_csv('wdbc.data',header=None)
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']
data.to_csv('labeledData.csv',header=headers,index=False)
data=pd.read_csv('labeledData.csv')
```

data.head()

[3]:
```
data.head()
```

[3]:
```
         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]
```

[4]:
```python
def diag(z):
    if z== 'M':
        return 1
    else:
        return 0

z=data['diagnosis'].apply(diag)
data.diagnosis=z
```

[6]:
```python
df = pd.DataFrame(data=data)
df=df.drop('id',axis=1)

x=df.drop('diagnosis',axis=1)
y=df['diagnosis']
```

[7]:
```python
x_scaled=x.apply(zscore)
x_scaled.describe()
```

[7]:
```
        mean_radius  mean_texture  mean_perimeter     mean_area
count  5.690000e+02  5.690000e+02    5.690000e+02  5.690000e+02  \
mean  -1.373633e-16  6.868164e-17   -1.248757e-16 -2.185325e-16
std    1.000880e+00  1.000880e+00    1.000880e+00  1.000880e+00
```

```
min   -2.029648e+00 -2.229249e+00   -1.984504e+00 -1.454443e+00
25%   -6.893853e-01 -7.259631e-01   -6.919555e-01 -6.671955e-01
50%   -2.150816e-01 -1.046362e-01   -2.359800e-01 -2.951869e-01
75%    4.693926e-01  5.841756e-01    4.996769e-01  3.635073e-01
max    3.971288e+00  4.651889e+00    3.976130e+00  5.250529e+00

       mean_smoothness  mean_compactness  mean_concavity  mean_concave points  \
count     5.690000e+02      5.690000e+02    5.690000e+02         5.690000e+02
mean     -8.366672e-16      1.873136e-16    4.995028e-17        -4.995028e-17
std       1.000880e+00      1.000880e+00    1.000880e+00         1.000880e+00
min      -3.112085e+00     -1.610136e+00   -1.114873e+00        -1.261820e+00
25%      -7.109628e-01     -7.470860e-01   -7.437479e-01        -7.379438e-01
50%      -3.489108e-02     -2.219405e-01   -3.422399e-01        -3.977212e-01
75%       6.361990e-01      4.938569e-01    5.260619e-01         6.469351e-01
max       4.770911e+00      4.568425e+00    4.243589e+00         3.927930e+00

       mean_symmetry  mean_fractal dimension  …  worst_radius  \
count   5.690000e+02            5.690000e+02  …  5.690000e+02
mean    1.748260e-16            4.745277e-16  … -8.241796e-16
std     1.000880e+00            1.000880e+00  …  1.000880e+00
min    -2.744117e+00           -1.819865e+00  … -1.726901e+00
25%    -7.032397e-01           -7.226392e-01  … -6.749213e-01
50%    -7.162650e-02           -1.782793e-01  … -2.690395e-01
75%     5.307792e-01            4.709834e-01  …  5.220158e-01
max     4.484751e+00            4.910919e+00  …  4.094189e+00

       worst_texture  worst_perimeter  worst_area  worst_smoothness  \
count   5.690000e+02     5.690000e+02  569.000000      5.690000e+02
mean    1.248757e-17    -3.746271e-16    0.000000     -2.372638e-16
std     1.000880e+00     1.000880e+00    1.000880      1.000880e+00
min    -2.223994e+00    -1.693361e+00   -1.222423     -2.682695e+00
25%    -7.486293e-01    -6.895783e-01   -0.642136     -6.912304e-01
50%    -4.351564e-02    -2.859802e-01   -0.341181     -4.684277e-02
75%     6.583411e-01     5.402790e-01    0.357589      5.975448e-01
max     3.885905e+00     4.287337e+00    5.930172      3.955374e+00

       worst_compactness  worst_concavity  worst_concave points  \
count       5.690000e+02     5.690000e+02          5.690000e+02
mean       -3.371644e-16     7.492542e-17          2.247763e-16
std         1.000880e+00     1.000880e+00          1.000880e+00
min        -1.443878e+00    -1.305831e+00         -1.745063e+00
25%        -6.810833e-01    -7.565142e-01         -7.563999e-01
50%        -2.695009e-01    -2.182321e-01         -2.234689e-01
75%         5.396688e-01     5.311411e-01          7.125100e-01
max         5.112877e+00     4.700669e+00          2.685877e+00

       worst_symmetry  worst_fractal dimension
```

```
count    5.690000e+02            5.690000e+02
mean     2.622390e-16           -5.744282e-16
std      1.000880e+00            1.000880e+00
min     -2.160960e+00           -1.601839e+00
25%     -6.418637e-01           -6.919118e-01
50%     -1.274095e-01           -2.164441e-01
75%      4.501382e-01            4.507624e-01
max      6.046041e+00            6.846856e+00

[8 rows x 30 columns]
```

[8]:
```
num_folds=10
kfold=KFold(n_splits=num_folds)
model=KNeighborsClassifier()
results=cross_val_score(model,x_scaled,y,cv=kfold)
print(results.mean())
```

```
0.9666040100250626
```

[9]:
```
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.
 ↪3,random_state=42)
```

[13]:
```
knn=KNeighborsClassifier(n_neighbors=5,weights='distance')
knn.fit(x_train,y_train)
```

[13]:
```
KNeighborsClassifier(weights='distance')
```

[14]:
```
predicted_labels=knn.predict(x_test)
knn.score(x_test,y_test)
```

[14]:
```
0.9590643274853801
```

[15]:
```
from sklearn import metrics
print('Confusion Matrix')
cm=metrics.confusion_matrix(y_test,predicted_labels,labels=[0,1])
df_cm=pd.DataFrame(cm,index=[i for i in [0,1]],
                   columns=[i for i in ['Predict 0','Predict 1']])
plt.figure(figsize=(7,5))
sns.heatmap(df_cm,annot=True,fmt='.5g',cmap='YlGn')
```

```
Confusion Matrix
```

[15]:
```
<Axes: >
```

[16]:
```
from sklearn.metrics import␣
↪roc_auc_score,roc_curve,classification_report,ConfusionMatrixDisplay


print(classification_report(y_test,predicted_labels))
```

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       108
           1       0.95      0.94      0.94        63

    accuracy                           0.96       171
   macro avg       0.96      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```

[17]:
```
from sklearn.model_selection import GridSearchCV
#Hyperparameters to be tuned
leaf_size=list(range(1,50))
n_neighbors=list(range(1,30))
p=[1,2]
```

```
hyperparameters=dict(leaf_size=leaf_size,n_neighbors=n_neighbors,p=p)
#create a new KNN object
knn_2=KNeighborsClassifier()
clf=GridSearchCV(knn_2,hyperparameters,cv=10)
#fit the model
best_model=clf.fit(x_scaled,y)
print('Best leaf_size:',best_model.best_estimator_.get_params()['leaf_size'])
print('Best p:',best_model.best_estimator_.get_params()['p'])
print('Best n_neighbors:',best_model.best_estimator_.
 ↪get_params()['n_neighbors'])
```

```
Best leaf_size: 1
Best p: 1
Best n_neighbors: 3
```

[18]:
```
y_pred=best_model.predict(x_test)
best_model.score(x_test,y_test)
```

[18]: 0.9824561403508771

[19]:
```
from sklearn import metrics
print('Confusion Matrix')
cm=metrics.confusion_matrix(y_test,y_pred,labels=[0,1])
df_cm=pd.DataFrame(cm,index=[i for i in [0,1]],
                   columns=[i for i in ['Predict 0','Predict 1']])
plt.figure(figsize=(7,5))
sns.heatmap(df_cm,annot=True,fmt='.5g',cmap='YlGn')
```

```
Confusion Matrix
```

[19]: <Axes: >

[20]:
```
false_negatives=np.logical_and(y_test!=y_pred,y_pred==0)
x_test[false_negatives]
```

[20]:
```
     mean_radius  mean_texture  mean_perimeter  mean_area  mean_smoothness
73     -0.092956     -0.814392       -0.063393  -0.201331         0.308838  \
255    -0.047513     -0.521181       -0.022203  -0.149284         0.942210
414     0.284783      2.448156        0.195281   0.183760        -0.936557

     mean_compactness  mean_concavity  mean_concave points  mean_symmetry
73           0.448373       -0.136966             0.045677      -0.546249  \
255          0.446478        0.114133             0.091333       0.351883
414         -1.104700       -0.526547            -0.555322       0.147430

     mean_fractal dimension  …  worst_radius  worst_texture
73                 0.405774  …      0.062293      -0.784455  \
255               -0.212302  …      0.025018      -0.587414
414               -1.397419  …      0.205179       1.829188

     worst_perimeter  worst_area  worst_smoothness  worst_compactness
73          0.090513   -0.119860          0.382749           0.635726  \
255         0.024984   -0.095952          0.825491           0.457607
```

```
414           0.084556     0.089332          -0.770135          -0.989865

     worst_concavity  worst_concave points  worst_symmetry
73          0.027401              0.360776       -0.504352  \
255         0.233695              0.347072        0.270565
414        -0.563654             -0.743914        0.537498

     worst_fractal dimension
73                  1.055903
255                -0.242489
414                -1.235541

[3 rows x 30 columns]
```

[21]:

```python
true_negatives=np.logical_and(y_test==y_pred,y_pred==0)
frames=[x_test[false_negatives],x_test[true_negatives]]
pred_neg=pd.concat(frames)
pred_neg
```

[21]:

```
     mean_radius  mean_texture  mean_perimeter  mean_area  mean_smoothness
73     -0.092956     -0.814392       -0.063393  -0.201331         0.308838  \
255    -0.047513     -0.521181       -0.022203  -0.149284         0.942210
414     0.284783      2.448156        0.195281   0.183760        -0.936557
204    -0.470694     -0.160486       -0.448110  -0.491999         0.234114
431    -0.490575     -0.374576       -0.432457  -0.532101         0.643316
..           …             …               …         …              …
426    -1.035883     -1.002884       -1.008296  -0.913779         0.128078
69     -0.382650     -0.651497       -0.436576  -0.433410         0.138753
542     0.174018      1.426574        0.112489   0.038995        -0.968582
176    -1.199475     -0.286147       -1.127336  -1.002515         0.044814
247    -0.351408     -1.205339       -0.289115  -0.405822        -0.623429

     mean_compactness  mean_concavity  mean_concave points  mean_symmetry
73           0.448373       -0.136966             0.045677      -0.546249  \
255          0.446478        0.114133             0.091333       0.351883
414         -1.104700       -0.526547            -0.555322       0.147430
204          0.027651       -0.109847            -0.276232       0.413949
431          0.516599       -0.142993            -0.539846      -0.002259
..                …               …                    …              …
426         -0.057631       -0.319515            -0.689709       0.413949
69          -0.985496       -0.656240            -0.523080      -0.809117
542         -0.610256       -0.599491            -0.481036       0.103619
176          0.474905        0.526062            -0.303315      -0.520693
247          0.573453        0.610180            -0.235219      -0.787211

     mean_fractal dimension  …  worst_radius  worst_texture
73                 0.405774  …      0.062293      -0.784455  \
```

```
255             -0.212302  …     0.025018     -0.587414
414             -1.397419  …     0.205179      1.829188
204              0.132176  …    -0.269040     -0.168905
431              1.165609  …    -0.701842     -0.450625
..                   …  …         …            …
426              0.900517  …    -0.857154     -0.668836
69              -0.888499  …    -0.581734     -0.963583
542             -0.850224  …     0.049868      1.076850
176              2.603060  …    -1.037316     -0.209616
247              0.183210  …    -0.389147     -1.299041

     worst_perimeter  worst_area  worst_smoothness  worst_compactness
73          0.090513   -0.119860          0.382749           0.635726  \
255         0.024984   -0.095952          0.825491           0.457607
414         0.084556    0.089332         -0.770135          -0.989865
204        -0.333935   -0.356299          0.448503          -0.104741
431        -0.525756   -0.641257          0.553709           0.054930
..               …           …                 …                  …
426        -0.770000   -0.773804          0.014527           0.288394
69         -0.643112   -0.572523         -0.121364          -1.168303
542         0.004134   -0.095249         -1.155891          -0.742153
176        -1.018414   -0.862051         -0.099446           0.259131
247        -0.067352   -0.424506         -0.305475           2.103300

     worst_concavity  worst_concave points  worst_symmetry
73          0.027401              0.360776       -0.504352  \
255         0.233695              0.347072        0.270565
414        -0.563654             -0.743914        0.537498
204        -0.024412             -0.199563        0.183204
431        -0.152986             -0.622863       -0.557739
..               …                     …              …
426         0.104162             -0.327467        0.192911
69         -0.807368             -0.849434       -0.837615
542        -0.532950             -0.077750       -0.289188
176         0.366586             -0.236107       -0.463908
247         2.401216              0.631809       -0.423463

     worst_fractal dimension
73                  1.055903
255                -0.242489
414                -1.235541
204                 0.196958
431                 0.534440
..                       …
426                 0.693484
69                 -1.099772
542                -0.797202
```

```
176                 1.787392
247                 1.876057

[111 rows x 30 columns]
```

[22]:
```
stacks=[y_test[false_negatives],y_test[true_negatives]]
y_labels=np.hstack(stacks)
y_labels.shape
print(y_labels)
```

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

[23]:
```
new_df=pd.DataFrame(data=pred_neg)
new_df['diagnosis']=y_labels
new_df.shape
new_df.head()
```

[23]:
```
     mean_radius  mean_texture  mean_perimeter  mean_area  mean_smoothness  \
73     -0.092956     -0.814392       -0.063393  -0.201331         0.308838
255    -0.047513     -0.521181       -0.022203  -0.149284         0.942210
414     0.284783      2.448156        0.195281   0.183760        -0.936557
204    -0.470694     -0.160486       -0.448110  -0.491999         0.234114
431    -0.490575     -0.374576       -0.432457  -0.532101         0.643316

     mean_compactness  mean_concavity  mean_concave points  mean_symmetry  \
73           0.448373       -0.136966             0.045677      -0.546249
255          0.446478        0.114133             0.091333       0.351883
414         -1.104700       -0.526547            -0.555322       0.147430
204          0.027651       -0.109847            -0.276232       0.413949
431          0.516599       -0.142993            -0.539846      -0.002259

     mean_fractal dimension  …  worst_texture  worst_perimeter  worst_area  \
73                 0.405774  …      -0.784455         0.090513   -0.119860
255               -0.212302  …      -0.587414         0.024984   -0.095952
414               -1.397419  …       1.829188         0.084556    0.089332
204                0.132176  …      -0.168905        -0.333935   -0.356299
431                1.165609  …      -0.450625        -0.525756   -0.641257

     worst_smoothness  worst_compactness  worst_concavity  \
73           0.382749           0.635726         0.027401
255          0.825491           0.457607         0.233695
414         -0.770135          -0.989865        -0.563654
204          0.448503          -0.104741        -0.024412
431          0.553709           0.054930        -0.152986
```

```
     worst_concave points  worst_symmetry  worst_fractal dimension  diagnosis
73               0.360776       -0.504352                 1.055903          1
255              0.347072        0.270565                -0.242489          1
414             -0.743914        0.537498                -1.235541          1
204             -0.199563        0.183204                 0.196958          0
431             -0.622863       -0.557739                 0.534440          0

[5 rows x 31 columns]
```

[24]:
```python
new_df['diagnosis'].value_counts()
```

[24]:
```
diagnosis
0    108
1      3
Name: count, dtype: int64
```

[25]:
```python
new_df_corr=new_df.corr()['diagnosis'].abs().sort_values(ascending=False)
new_df_corr
```

[25]:
```
diagnosis                  1.000000
worst_area                 0.316577
worst_radius               0.289529
worst_perimeter            0.286102
SE_area                    0.230159
mean_area                  0.229837
mean_perimeter             0.216750
mean_radius                0.211266
worst_concave points       0.169396
mean_concave points        0.169167
SE_radius                  0.131486
worst_compactness          0.130878
mean_concavity             0.115360
mean_compactness           0.113136
SE_perimeter               0.112116
mean_texture               0.108400
worst_concavity            0.087137
worst_smoothness           0.081663
worst_texture              0.081404
mean_fractal dimension     0.080556
SE_smoothness              0.070546
worst_symmetry             0.068493
SE_texture                 0.062043
SE_fractal dimension       0.059387
mean_smoothness            0.051853
SE_concave points          0.031789
worst_fractal dimension    0.030647
mean_symmetry              0.024369
```

```
SE_compactness              0.005433
SE_symmetry                 0.002118
SE_concavity                0.001798
Name: diagnosis, dtype: float64
```

[26]:
```
features=new_df_corr[new_df_corr>0.2].index.to_list()[1:]
features
```

[26]:
```
['worst_area',
 'worst_radius',
 'worst_perimeter',
 'SE_area',
 'mean_area',
 'mean_perimeter',
 'mean_radius']
```

[27]:
```
from sklearn.linear_model import LinearRegression
def calculate_vif(df, features):
    vif, tolerance = {}, {}
    # all the features that you want to examine
    for feature in features:
        # extract all the other features you will regress against
        x = [f for f in features if f != feature]
        x, y = df[x], df[feature]
        # extract r-squared from the fit
        r2 = LinearRegression().fit(x, y).score(x, y)

        # calculate tolerance
        tolerance[feature] = 1 - r2
        # calculate VIF
        vif[feature] = 1/(tolerance[feature])
    # return VIF DataFrame
    return pd.DataFrame({'VIF': vif, 'Tolerance': tolerance})
calculate_vif(new_df,features)
```

[27]:

	VIF	Tolerance
worst_area	245.432662	0.004074
worst_radius	225.617968	0.004432
worst_perimeter	74.357191	0.013449
SE_area	1.427665	0.700444
mean_area	368.318175	0.002715
mean_perimeter	398.964991	0.002506
mean_radius	692.776191	0.001443

[28]:
```
features=['worst_radius','SE_area','worst_concave points']
calculate_vif(new_df,features)
```

[28]:

```
                           VIF  Tolerance
worst_radius          1.544534   0.647445
SE_area               1.094041   0.914042
worst_concave points  1.432551   0.698055
```

[31]:

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
x=new_df.loc[:,features]
y=new_df.loc[:,'diagnosis']
random_state=42
x_train,x_test,y_train,y_test=\
train_test_split(x,y,test_size=0.3,shuffle=True,random_state=random_state)
knn_n=KNeighborsClassifier()
knn_n.fit(x_train,y_train)

knn_n.score(x_test,y_test)
```

[31]: 0.9705882352941176

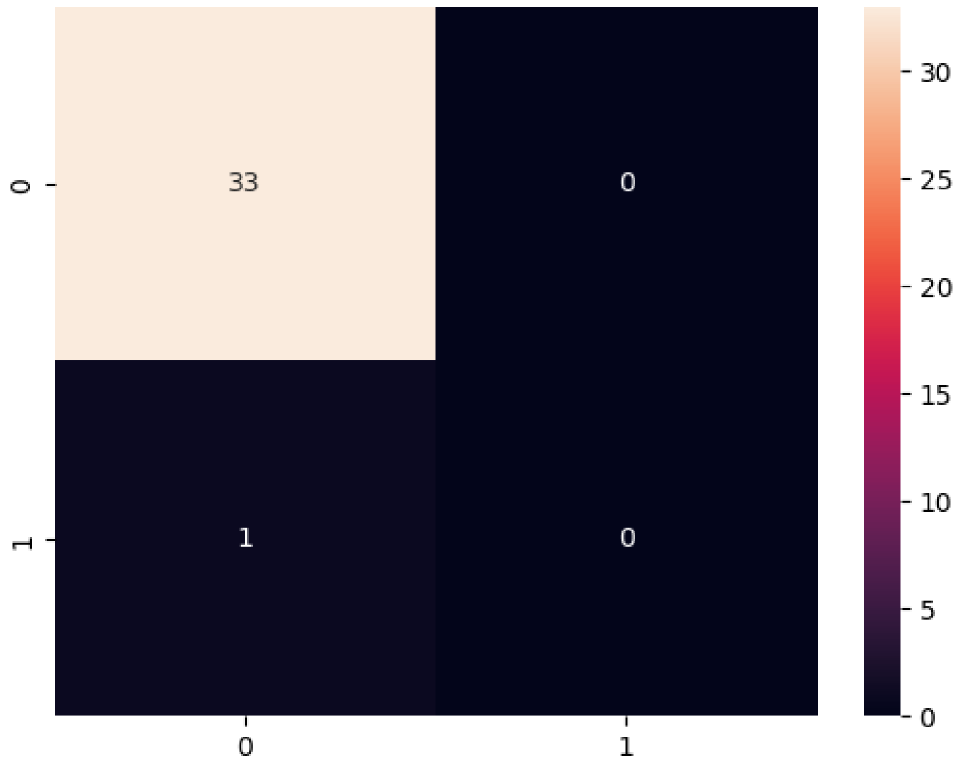
[32]:

```python
y_pred=knn_n.predict(x_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        33
           1       0.00      0.00      0.00         1

    accuracy                           0.97        34
   macro avg       0.49      0.50      0.49        34
weighted avg       0.94      0.97      0.96        34


c:\Users\sheel\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\sheel\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\sheel\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[34]:
```
cm=metrics.confusion_matrix(y_test,knn_n.predict(x_test))
sns.heatmap(cm,annot=True,fmt='d')
```

[34]: <Axes: >



[35]:
```
leaf_size=list(range(1,50))
n_neighbors=list(range(1,30))
p=[1,2]
hyperparameters=dict(leaf_size=leaf_size,n_neighbors=n_neighbors,p=p)
#create a new KNN object
knn_n_tuned=KNeighborsClassifier()
clf=GridSearchCV(knn_n_tuned,hyperparameters,cv=10)
#fit the model
best_model=clf.fit(x,y)
print('Best leaf_size:',best_model.best_estimator_.get_params()['leaf_size'])
print('Best p:',best_model.best_estimator_.get_params()['p'])
print('Best n_neighbors:',best_model.best_estimator_.
 ↪get_params()['n_neighbors'])
```

```
c:\Users\sheel\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\model_selection\_split.py:700: UserWarning: The least populated
class in y has only 3 members, which is less than n_splits=10.
```

```
  warnings.warn(
```

```
Best leaf_size: 1
Best p: 1
Best n_neighbors: 4
```

[36]:
```python
y_pred=best_model.predict(x_test)
best_model.score(x_test,y_test)
```

[36]: 0.9705882352941176

[37]:
```python
from sklearn import metrics
print('Confusion Matrix')
cm=metrics.confusion_matrix(y_test,y_pred,labels=[0,1])
df_cm=pd.DataFrame(cm,index=[i for i in [0,1]],
                   columns=[i for i in ['Predict 0','Predict 1']])
plt.figure(figsize=(7,5))
sns.heatmap(df_cm,annot=True,fmt='.5g',cmap='YlGn')
```

```
Confusion Matrix
```

[37]: <Axes: >

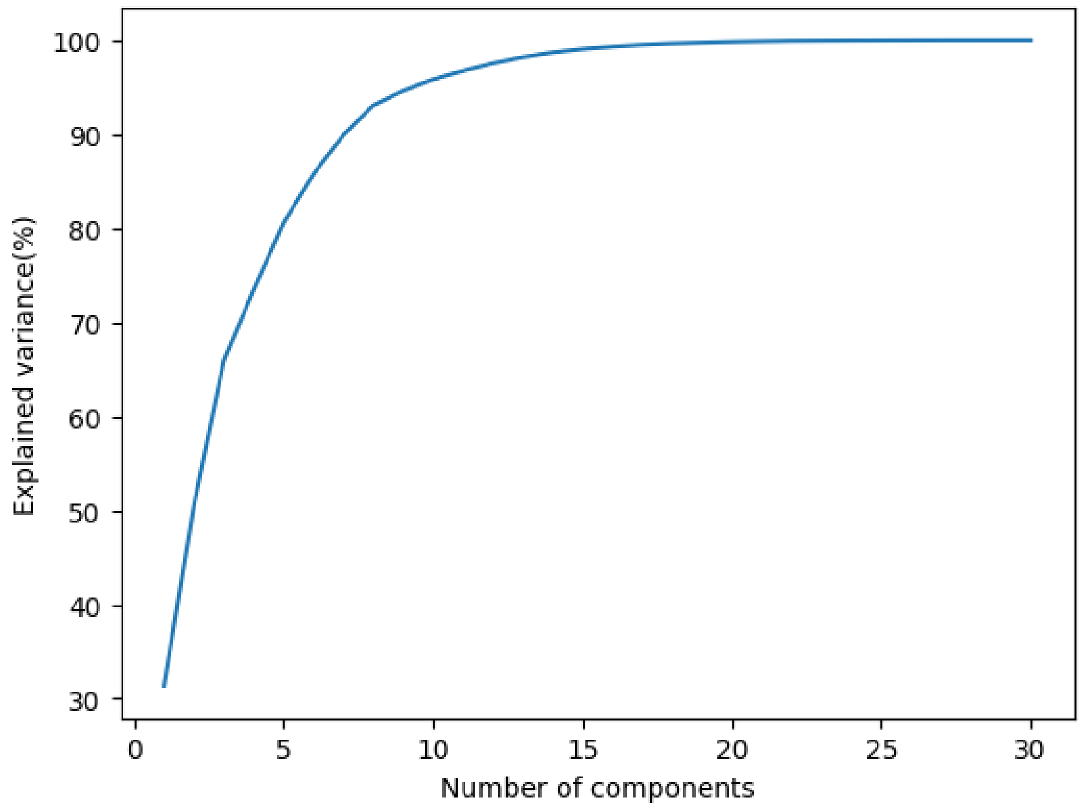
[38]:
```
x=new_df.drop('diagnosis',axis=1)
y=new_df['diagnosis']
```

[68]:
```
from sklearn.decomposition import PCA
components=None
pca_n=PCA(n_components=components)
pca_n.fit(x)
print('Cumulative Variances Percentage:')
print(pca_n.explained_variance_ratio_.cumsum()*100)
```

```
Cumulative Variances Percentage:
[ 31.33946457  50.48852305  65.90396901  73.46764938  80.57795264
  85.77634153  89.91095462  93.0615124   94.63601746  95.83543616
  96.7530737   97.55365698  98.19564115  98.70547047  99.06197239
  99.32088691  99.5181861   99.65640052  99.75321254  99.83023926
  99.89230473  99.94331584  99.96252949  99.97651498  99.98519981
  99.99240865  99.99766401  99.99947357  99.99988405 100.        ]
```

[69]:
```
components=len(pca_n.explained_variance_ratio_)\
    if components is None else components
plt.plot(range(1,components+1),
         np.cumsum(pca_n.explained_variance_ratio_*100))
plt.xlabel('Number of components')
plt.ylabel('Explained variance(%)')
```

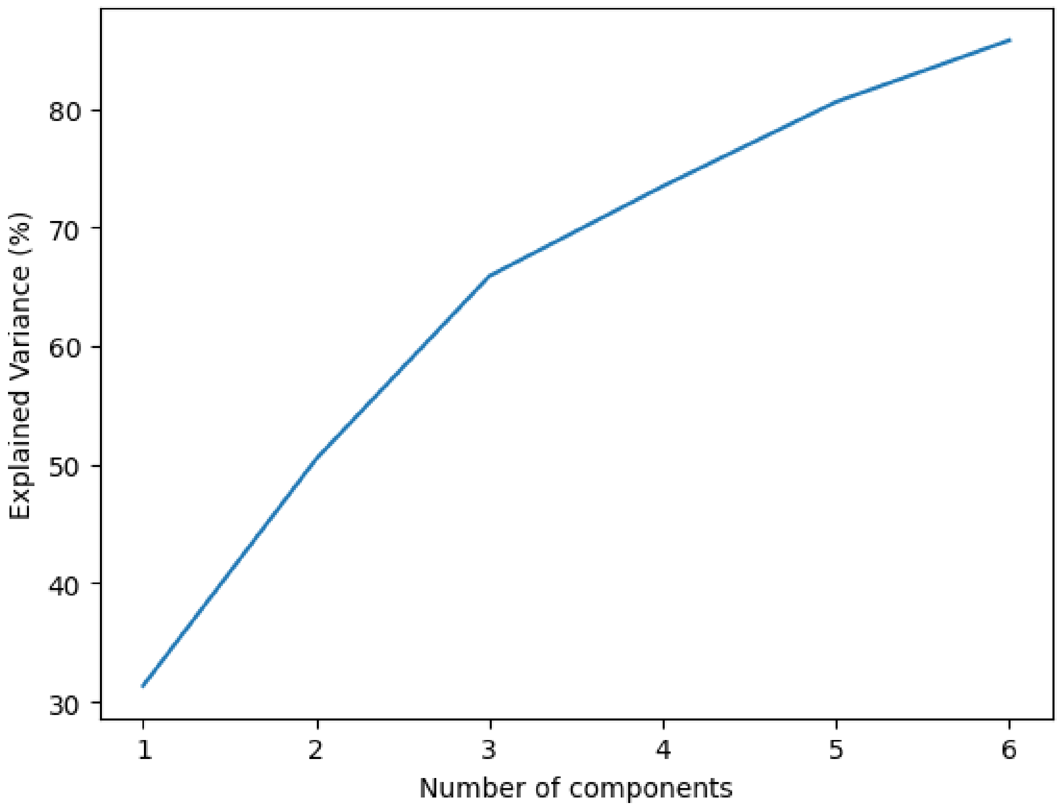
[69]:
```
Text(0, 0.5, 'Explained variance(%)')
```



[70]:
```
from sklearn.decomposition import PCA
pca_n=PCA(n_components=0.85)
pca_n.fit(x)
print('Cumulative Variances (Percentage):')
print(np.cumsum(pca_n.explained_variance_ratio_*100))
components=len(pca_n.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):
[31.33946457 50.48852305 65.90396901 73.46764938 80.57795264 85.77634153]
Number of components:6
```

[70]:
```
Text(0, 0.5, 'Explained Variance (%)')
```



[71]:
```python
pca_n_components=abs(pca_n.components_)
print(pca_n_components)
```

```
[[0.0398405  0.09848313 0.02644045 0.03037346 0.24246847 0.19533846
  0.17528555 0.10540765 0.16771765 0.28742308 0.08342074 0.01948502
  0.07245904 0.03149686 0.18141487 0.33345417 0.36873073 0.30967724
  0.20428427 0.360185   0.02711115 0.10088855 0.01778323 0.02117902
  0.16145573 0.15390203 0.20569824 0.12655342 0.10963287 0.21160416]
 [0.18700926 0.22039956 0.1842136  0.13469716 0.01041502 0.09965415
  0.09907305 0.10504584 0.13107073 0.11301301 0.10972896 0.5423484
  0.08349876 0.02407898 0.32885035 0.02757957 0.09028853 0.07576705
  0.36647176 0.09367006 0.15775297 0.17777402 0.16160044 0.10584086
  0.04324737 0.18280391 0.2028868  0.2225959  0.00422999 0.06405447]
 [0.09611995 0.37693681 0.09449309 0.07482681 0.39555849 0.01340619
  0.11607581 0.01434893 0.29417044 0.11728469 0.01679698 0.24999813
  0.02058442 0.0254976  0.1705051  0.22940584 0.27362205 0.14082978
  0.05207832 0.18735391 0.07237197 0.3262301  0.07477771 0.05367087
  0.31253305 0.07681412 0.16254331 0.05418525 0.17998322 0.00067994]
 [0.05126195 0.44620084 0.04654946 0.04101006 0.12717303 0.07452738
  0.02777023 0.01965175 0.13403923 0.09298639 0.10726061 0.07481549
  0.0897387  0.06185864 0.03337895 0.12872113 0.00603186 0.16391903
```

```
  0.10813163 0.21785043 0.00531477 0.51308269 0.0020477  0.0075484
  0.33242892 0.17168171 0.17645384 0.09373479 0.35747789 0.20290172]
 [0.22029273 0.08949819 0.21522999 0.15523463 0.44345821 0.11181559
  0.03538345 0.2121461  0.11407049 0.14012456 0.12466565 0.32766419
  0.10050078 0.08687796 0.23090245 0.06635451 0.17747266 0.21711192
  0.16741592 0.10367328 0.16689182 0.10681268 0.15301483 0.10967127
  0.253817   0.03041527 0.08257865 0.15077911 0.26490117 0.17180321]
 [0.16631197 0.08509915 0.15812546 0.12469957 0.01808657 0.0040162
  0.01912854 0.09094093 0.58229951 0.20969123 0.16860393 0.00593632
  0.14634841 0.09760257 0.25772033 0.02193658 0.00398886 0.15185195
  0.3105958  0.10832614 0.133729   0.00761727 0.12481949 0.09250251
  0.2423356  0.05500226 0.0300897  0.05208917 0.3422298  0.2244229 ]]
```

[72]:

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

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

    # print the top 4 feature names
    new_df_2=new_df.drop('diagnosis',axis=1)
    print(f'Component {row}: {new_df_2.columns[indices].to_list()}')
```

```
Top 4 most important features in each component
================================================
Component 0: ['SE_concavity', 'SE_fractal dimension', 'SE_compactness',
'SE_concave points']
Component 1: ['SE_texture', 'SE_symmetry', 'SE_smoothness', 'worst_concave
points']
Component 2: ['mean_smoothness', 'mean_texture', 'worst_texture',
'worst_smoothness']
Component 3: ['worst_texture', 'mean_texture', 'worst_symmetry',
'worst_smoothness']
Component 4: ['mean_smoothness', 'SE_texture', 'worst_symmetry',
'worst_smoothness']
Component 5: ['mean_symmetry', 'worst_symmetry', 'SE_symmetry', 'SE_smoothness']
```

[73]:

```python
x_pca=pca_n.transform(x)
print(x_pca.shape)
print(x_pca)
```

```
(111, 6)
[[ 6.79380272e-01 -2.88709733e+00 -3.16523777e-01 -5.74445110e-01
   6.73650305e-01 -5.78048064e-01]
 [ 1.51906853e+00 -2.01913988e+00 -5.78533787e-01 -7.42637278e-01
```

```
  1.38409723e+00  1.00634290e+00]
 [-1.62602449e+00  1.24562031e+00  2.32585701e+00 -1.61091313e+00
   1.13407345e+00  2.69293962e+00]
 [ 8.25891520e-01 -8.58639487e-01 -3.29820027e-01 -9.19320577e-01
   5.22643726e-01  3.70025600e-01]
 [ 2.36741805e+00  8.80069983e-02 -1.19504568e-01 -2.68335119e-01
   5.52500377e-01 -1.02666536e+00]
 [ 2.23261184e+00  5.56892274e-01 -1.12633283e-01  1.10267613e+00
   6.62406810e-01 -1.19395924e+00]
 [ 3.29813391e+00 -2.14184164e+00 -3.74076747e-01 -1.64093325e+00
   5.13524326e-01  2.83833173e-01]
 [-2.50668300e+00 -1.89392057e+00  6.33788717e-01  6.94382387e-01
  -1.37272419e+00  7.80914941e-01]
 [-2.39297206e+00  8.23277608e-02  1.37058052e+00 -1.38032685e+00
   8.67299428e-01 -1.00756230e-01]
 [-2.89307455e+00 -1.01618824e+00  9.76599432e-01  4.44813981e-01
   3.18357333e-01  8.40149981e-01]
 [-1.56419344e+00 -2.38825496e+00 -1.25453744e+00  1.10218088e-01
  -7.71051954e-01 -1.74287460e-01]
 [-1.86535230e+00 -1.83448515e+00 -1.99086673e-01  1.67029671e+00
   4.38909235e-01  4.83478740e-01]
 [-5.60119854e-01  2.69534543e+00 -8.11731050e-01  5.33416129e-01
   7.63942564e-01 -9.36234887e-01]
 [ 7.32930289e-01  2.46590747e+00 -3.70114636e+00 -4.27699094e-01
  -1.05927882e+00 -2.07916822e+00]
 [-1.63261963e-01  4.53119701e-01 -8.38424639e-01 -7.57869308e-01
   8.87707214e-01  3.64260674e-01]
 [-2.47844529e+00  1.02122091e+00  3.12673228e-01 -3.97575204e-01
  -9.43870654e-01 -1.14985468e+00]
 [-1.45976810e+00  1.98501803e+00  1.45333819e+00 -6.49161572e-01
  -1.02741990e+00  1.67008028e+00]
 [ 2.41623969e+00  1.44999435e-01 -3.47160952e+00  1.43905107e+00
   1.61766361e+00  8.70819564e-01]
 [-8.23173598e-01 -2.55787507e+00  4.21768554e-02  9.13620853e-01
  -6.78663523e-01  5.44746419e-02]
 [ 7.35117812e-01  2.17062970e+00  2.28147034e+00 -1.32881184e+00
   1.29483561e+00 -1.46696774e+00]
 [-7.81663960e-01 -3.08126562e-01 -6.96029212e-02 -3.97481979e-01
   4.35593990e-01  1.79830835e-01]
 [ 5.54980428e-01  9.24422263e-01  7.69335237e-01 -7.29398489e-01
  -1.02186126e+00  1.29323148e+00]
 [-1.48944868e+00 -1.32754751e+00 -1.17043858e+00  9.61986570e-01
  -3.84308493e-01 -8.19661292e-01]
 [ 1.64364400e+00  3.18019847e+00 -2.30805748e+00 -3.27491553e-01
   4.09059730e-01  1.41056076e+00]
 [-1.79217710e+00 -2.22568727e+00 -7.46807989e-01 -2.47881184e-03
  -1.11909902e-01 -1.40923604e-01]
 [ 4.10569269e-01  9.96328700e-01 -1.74085352e+00 -5.55211399e-01
```

```
 -4.26230202e-01 -3.76457219e-01]
[-5.79587471e-01  7.83340461e-01 -1.15168822e+00 -3.57192096e-02
 -4.42238373e-01  1.17309108e+00]
[-1.51791351e-01  6.79210796e-01 -6.73764664e-01  1.78654170e+00
 -5.07180576e-01  2.10293857e+00]
[-1.40334717e+00 -4.30646025e-01  1.37745132e+00 -3.87929622e-01
  7.99114580e-01  9.60467751e-01]
[ 4.01975040e-01  1.54670957e+00 -8.14597906e-01 -6.25386728e-02
 -4.81783422e-01  9.03465162e-01]
[ 7.06102381e-01 -1.40161946e+00 -1.28902766e+00 -2.17059150e-01
  4.25466658e-01  1.31419208e-01]
[-1.15405019e+00 -7.72545233e-01  2.56041876e-01 -2.03415481e-01
 -3.60445027e-01 -8.21579923e-01]
[ 1.24255186e+00  2.17576272e+00 -1.36286035e+00  1.46450261e+00
  3.36557191e+00  7.78254365e-01]
[-2.33518823e-01 -3.07982203e-01  5.05333701e-03 -1.90625638e+00
  1.68610703e-01 -1.77071762e+00]
[ 4.05919267e+00  1.96501925e+00 -4.47266644e+00 -8.87174177e-01
  7.83495481e-01 -2.31407306e-01]
[-1.93834510e+00  4.98882283e+00  1.43575061e+00 -6.21243068e-01
  3.34758127e-01 -1.73645044e-01]
[-2.21713725e-02 -9.48762911e-01 -6.41723037e-01 -2.23262954e+00
 -1.31502911e-01 -3.88300604e-01]
[ 8.95216822e-01 -1.35359207e+00  2.41967013e+00  1.99224210e+00
 -1.21358537e-02 -3.65073990e-01]
[ 1.98009260e+00 -1.09018551e+00  2.56990558e-01 -2.87132864e+00
 -6.21801133e-01  6.07667347e-01]
[ 3.11867077e+00  4.21819952e-01 -2.96673548e-01  2.26311029e+00
  3.64900097e+00  6.00991931e-01]
[ 3.76443191e+00  1.73022531e-01 -1.74310760e+00  8.11083841e-01
 -3.83055869e-01 -5.99615223e-01]
[-1.98509099e+00 -1.85882597e+00  8.35566721e-01  2.80054516e-01
  1.60329013e-01 -7.13157459e-01]
[ 3.16140562e+00  7.77078529e-01 -3.72220854e-01  5.16190379e-01
  8.22508841e-02 -1.27736393e+00]
[-3.73701711e-01 -1.32929089e+00 -1.85628481e+00  6.37534176e-01
 -5.20451836e-01 -5.99292735e-01]
[-5.27087839e-01 -8.81967884e-01 -8.91742984e-02 -3.67842718e-01
  3.52870547e-01  6.16584977e-02]
[-2.73883385e+00 -5.44856594e-01  2.38885384e+00 -7.02554571e-01
 -1.65806150e-02  7.62613495e-01]
[-2.12164396e+00  7.78417379e-01  3.74779836e-01  2.57998419e-02
  1.19396696e-01 -6.34656052e-01]
[-4.87769262e-01 -1.20670086e+00  2.13249309e+00 -1.72075032e+00
 -3.95774540e-01 -6.14675815e-01]
[-1.01216906e+00  1.53590670e+00 -7.23168270e-01 -2.82395580e-01
  3.54212262e-01  3.83202968e-01]
[ 1.60216285e+00 -1.39949506e+00  1.45667332e-01 -8.67816376e-01
```

```
 -9.25771373e-01  6.86688765e-01]
[-2.08288319e+00 -3.49464513e-01  9.03331798e-01  7.02218778e-01
  4.85922353e-01  4.20235811e-01]
[-2.68970707e-01  3.82393634e+00 -3.70718823e-01 -1.37438914e+00
 -1.80898313e+00 -1.30129113e+00]
[ 4.97088299e-01 -2.40362636e+00 -1.55480106e+00 -4.63497240e-01
  1.13021708e-01  2.20618243e-02]
[-1.63255102e+00 -1.86962967e-01  1.07962990e+00 -3.09681830e-01
  1.49834101e+00 -6.02806845e-01]
[-1.98567517e+00 -6.65546306e-02  3.56701696e-01  1.51899632e+00
 -8.19069820e-01 -1.07744947e+00]
[ 2.06636906e+00 -1.03856985e+00 -2.50188168e+00 -9.32690570e-01
  6.36045484e-01 -5.92052988e-03]
[-2.24367031e+00  5.94376786e-03 -1.00793798e+00  5.65294030e-01
 -1.15854215e+00 -1.96525170e-01]
[ 9.56152948e-01 -3.39578026e-01  3.88074058e+00  5.78105741e-01
 -9.90773725e-01 -1.17800840e+00]
[ 3.02975331e-01 -1.72879817e+00 -1.12694062e+00  5.31212165e-02
  1.26646055e+00 -1.37115744e+00]
[-1.58902730e-01 -3.53328520e-01 -9.82977941e-01 -8.05746757e-01
 -3.24022254e-01  8.22525700e-01]
[ 5.95050648e-02 -2.34671895e+00 -1.02118011e+00 -2.16300804e+00
  6.46891627e-01  5.87665774e-03]
[ 3.40520866e-02 -2.40564768e+00  3.44764140e-01  5.76362110e-01
  1.08705905e+00 -2.29998382e-02]
[-4.15929370e+00  4.41298784e+00  3.97562967e+00 -4.14060079e-02
  1.64554409e+00 -1.58093422e+00]
[-3.22203964e-01  3.28123045e+00 -1.67167185e+00 -4.65687468e-01
  5.82280119e-01  7.75787135e-01]
[ 1.71364729e+00  1.91491959e+00 -1.10191540e+00  4.64267181e-01
  2.70910730e-01 -1.58008972e+00]
[-7.63890825e-01 -7.16569500e-02 -2.22999811e+00  9.55302602e-01
 -4.11898715e-01 -1.10253082e-01]
[-1.39709296e+00  5.02469148e-02 -2.83743885e+00  9.79138848e-01
 -2.30469065e+00 -3.98442805e-01]
[-8.33369333e-01  2.35303723e+00  7.26808665e-01 -1.00236438e+00
 -1.85169495e-01  8.37041446e-01]
[ 2.53582308e-01  3.83454208e-01  1.24828664e-01 -2.13054751e-01
  1.68701355e+00 -4.26822183e-01]
[ 1.17075090e+00  7.59717721e-01  8.28953183e-01 -7.12237270e-01
  1.37994680e+00 -1.61923711e+00]
[ 6.23577104e-01 -2.72612441e+00 -2.68804320e-02  2.41052472e-01
  1.08583758e+00  4.10214203e-01]
[-2.24172482e+00 -2.08897589e+00  1.07865820e+00  3.27091477e-01
  5.86319687e-01  3.58200437e-02]
[-8.93660002e-02 -6.98613881e-01 -1.18958602e+00  7.17170538e-02
  2.16947890e-01 -1.85030962e-01]
[-8.05646594e-01  3.80594125e-01  4.36785748e+00 -1.31726225e+00
```

```
  1.92274022e+00 -7.11297218e-01]
[ 1.18174511e+01 -1.31641061e+00  3.29112314e+00 -1.71187248e+00
 -2.99942108e+00  4.63763668e-01]
[ 2.41733207e+00  2.83559616e+00 -5.02347899e-02  1.58676015e+00
 -2.01990673e+00  1.68357776e+00]
[-7.56797748e-01  1.43762990e+00 -9.35740574e-02 -2.68730252e+00
  8.70543382e-02  1.28620451e-01]
[ 1.71342276e+00  2.53545888e+00 -3.24813806e+00  4.18763730e-01
 -1.11819118e+00  2.65591978e+00]
[ 7.63764499e+00  9.01358944e-01  4.87933902e+00  3.74286489e+00
  1.41603685e-01  9.71725130e-01]
[-1.22608760e+00 -9.14468743e-01 -6.34055870e-01  3.42452692e-01
 -1.52020241e-01 -1.09674657e-01]
[-6.55383548e-01 -5.95061103e-01 -2.24987507e-01 -5.99381178e-01
 -7.43498258e-01  6.22963492e-01]
[ 7.62133676e-01  7.47585201e-02  1.05656710e+00 -1.29241226e+00
  2.54912796e-01  3.91706708e-01]
[ 3.19241849e-01  1.05274780e+00 -6.19281355e-01 -5.50047267e-01
 -1.03705396e+00  7.06535811e-01]
[-2.02073677e+00 -1.74527081e+00  2.20633415e-02  2.12317235e-01
 -7.03039466e-01 -6.76125567e-02]
[-2.69406607e+00 -1.98569492e+00  1.39946070e+00  3.68225078e-01
 -9.53125377e-01  5.33541127e-01]
[-1.35936036e+00  7.37708348e-01 -5.29058966e-01 -7.29598946e-01
 -6.75660882e-01  9.27567211e-01]
[-1.76373090e+00 -3.61954446e+00  2.02400031e-01  1.31615054e+00
  2.45087755e-01 -4.64815426e-01]
[ 5.49230609e-01 -1.59526643e+00  3.99414564e-01 -1.09348194e+00
  1.58511370e+00  1.89518920e-01]
[ 1.02244375e+00  3.68672540e+00 -6.44111823e-01 -5.59897858e-01
  1.81130239e+00 -6.70542646e-01]
[ 2.82695810e+00 -2.08260728e+00 -4.28459898e-01  4.30326637e-01
  9.71615726e-01  2.03460395e+00]
[-7.33826086e-01 -4.30963429e-02 -9.82874330e-01 -7.89659290e-01
  3.11839825e-01 -4.11850926e-01]
[-4.17219399e-01 -9.19703085e-01 -2.33212076e+00  6.11457650e-01
 -1.07268250e+00 -5.41697625e-01]
[-2.56703785e+00 -1.84816806e+00  2.64512211e-01  2.22055175e+00
 -1.44444819e+00 -6.77421902e-01]
[-3.39869707e-01 -3.15957821e-01  1.50865862e+00  5.95074536e-01
 -1.34069392e+00  1.66621427e+00]
[-2.26548812e+00 -1.52439165e+00 -1.38308902e-01  1.90275116e+00
  2.15586228e-01 -8.05907590e-01]
[-1.58069000e+00  9.67799110e-02 -5.71846807e-01  2.18660593e+00
 -1.61938198e-01  4.35974849e-01]
[-6.36816283e-01  1.22448180e+00 -1.39228199e+00 -3.95284633e-02
 -1.82568658e+00 -8.57209719e-01]
[-9.74493147e-01 -5.52486935e-01  4.17440248e-02 -1.61725854e-01
```

```
   1.28225399e+00  3.13668238e-01]
 [-2.96408173e+00  3.63194678e-01  1.82974894e+00 -1.36171255e+00
  -1.80864843e+00  6.84679931e-01]
 [-3.16837555e+00  2.73991162e+00  1.35115097e+00 -1.27187874e-01
  -2.57016829e+00 -2.28173586e-01]
 [-1.49634875e+00 -1.81986558e+00  6.88524275e-02  7.78394969e-01
   5.53795177e-01  2.31726061e-01]
 [-2.18507886e+00 -1.36175843e-03 -2.05819797e-01  1.20104354e+00
  -1.12341206e+00 -7.58108399e-01]
 [-2.88895866e+00  5.34227551e+00  2.06244541e+00  2.54403269e+00
  -5.23271239e-01  4.33608633e-01]
 [ 1.61470844e+00 -6.85284997e-01  7.45673074e-02 -8.62906110e-01
  -2.10534811e+00 -5.43873379e-01]
 [-6.54499854e-01  1.89040218e+00 -2.08689907e+00  4.28101468e-01
  -2.47797293e-02 -1.17009678e+00]
 [ 1.57963274e+00  7.96798845e-02  1.89851813e+00  1.46559804e+00
   7.44488404e-01 -1.23382452e-01]
 [ 2.30492267e+00 -4.07136659e-01 -3.43221876e-01 -1.31080916e-01
  -8.60277481e-01 -3.21279585e-01]
 [-1.38323768e+00 -4.40290397e-01 -8.87906992e-01  9.14557861e-01
   5.60739649e-01 -2.11255899e-01]
 [-1.48297414e+00 -3.37561350e-01  2.05073105e+00 -8.36312179e-01
   8.97611411e-01  1.65133179e+00]
 [ 7.99784675e+00  2.32656386e+00  2.97803726e+00  1.46993253e+00
  -7.81221387e-01 -1.70249199e+00]
 [ 3.73740870e+00 -3.98696567e+00  1.82608136e+00 -4.27484149e-03
  -1.20194667e+00 -1.24408201e+00]]
```

[74]:
```
x_train,x_test,y_train,y_test=train_test_split(x_pca,y,test_size=0.
↪3,random_state=42)
```

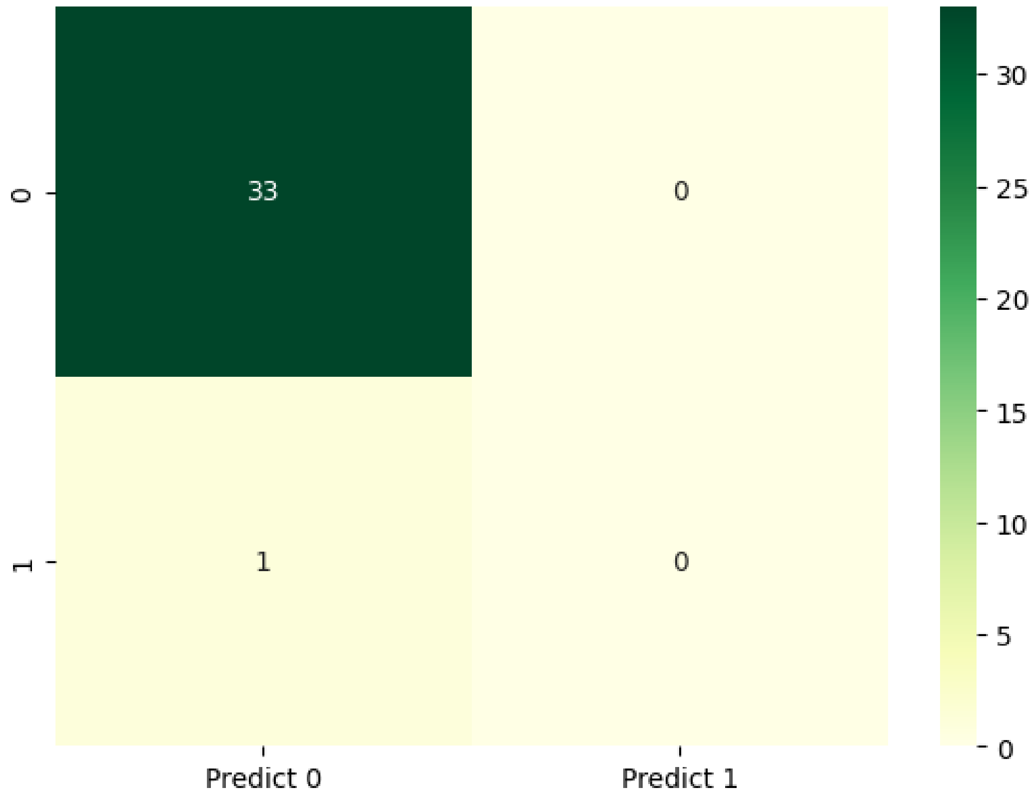
[75]:
```
knn_pca=KNeighborsClassifier(n_neighbors=5,weights='distance')
knn_pca.fit(x_train,y_train)

y_pred=knn_pca.predict(x_test)
knn_pca.score(x_test,y_test)
```

[75]:
```
0.9705882352941176
```

[76]:
```
from sklearn import metrics
print('Confusion Matrix')
cm=metrics.confusion_matrix(y_test,y_pred,labels=[0,1])
df_cm=pd.DataFrame(cm,index=[i for i in [0,1]],
                   columns=[i for i in ['Predict 0','Predict 1']])
plt.figure(figsize=(7,5))
sns.heatmap(df_cm,annot=True,fmt='.5g',cmap='YlGn')
```

```
Confusion Matrix
```

[76]: <Axes: >



[77]:
```
leaf_size=list(range(1,50))
n_neighbors=list(range(1,30))
p=[1,2]
hyperparameters=dict(leaf_size=leaf_size,n_neighbors=n_neighbors,p=p)
#create a new KNN object
knn_pca_tuned=KNeighborsClassifier()
clf=GridSearchCV(knn_pca_tuned,hyperparameters,cv=10)
#fit the model
best_model=clf.fit(x_pca,y)
print('Best leaf_size:',best_model.best_estimator_.get_params()['leaf_size'])
print('Best p:',best_model.best_estimator_.get_params()['p'])
print('Best n_neighbors:',best_model.best_estimator_.
 ↪get_params()['n_neighbors'])
```

```
c:\Users\sheel\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\model_selection\_split.py:700: UserWarning: The least populated
class in y has only 3 members, which is less than n_splits=10.
  warnings.warn(
```

```
Best leaf_size: 1
```

```
Best p: 1
Best n_neighbors: 2
```

[78]:
```
y_pred=best_model.predict(x_test)
best_model.score(x_test,y_test)
```

[78]:
```
0.9705882352941176
```

[79]:
```
print('Confusion Matrix')
cm=metrics.confusion_matrix(y_test,y_pred,labels=[0,1])
df_cm=pd.DataFrame(cm,index=[i for i in [0,1]],
                   columns=[i for i in ['Predict 0','Predict 1']])
plt.figure(figsize=(7,5))
sns.heatmap(df_cm,annot=True,fmt='.5g',cmap='YlGn')
```

```
Confusion Matrix
```

[79]:
```
<Axes: >
```