12-knn-classifier-breast-cancer-1

September 20, 2025

1 Breast Cancer - kNN Classifier

In kNN classification, the output is a class membership. The given data point is classified based on the majority of type of its neighbours. The data point is assigned to the most frequent class among its k nearest neighbours.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df = pd.read_csv(r"D:\Naresh It Classes\4. September\12- LGBM, Cross␣
↪Validation, AUC, ROC\kNN Classifier Project\breast-cancer.txt")
```

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

[3]:

	1000025	5	1	1.1	1.2	2	1.3	3	1.4	1.5	2.1
0	1002945	5	4	4	5	7	10	3	2	1	2
1	1015425	3	1	1	1	2	2	3	1	1	2
2	1016277	6	8	8	1	3	4	3	7	1	2
3	1017023	4	1	1	3	2	1	3	1	1	2
4	1017122	8	10	10	8	7	10	9	7	1	4

[5]:
```
df.shape
```

[5]:
```
(698, 11)
```

[6]:
```
col_names = ['Id', 'Clump_thickness', 'Uniformity_Cell_Size',␣
↪'Uniformity_Cell_Shape', 'Marginal_Adhesion',
             'Single_Epithelial_Cell_Size', 'Bare_Nuclei', 'Bland_Chromatin',␣
↪'Normal_Nucleoli', 'Mitoses', 'Class']

df.columns = col_names

df.columns
```

[6]:
```
Index(['Id', 'Clump_thickness', 'Uniformity_Cell_Size',
       'Uniformity_Cell_Shape', 'Marginal_Adhesion',
       'Single_Epithelial_Cell_Size', 'Bare_Nuclei', 'Bland_Chromatin',
       'Normal_Nucleoli', 'Mitoses', 'Class'],
      dtype='object')
```

[7]:
```
df.head()
```

[7]:

	Id	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bland_Chromatin	Normal_Nucleoli	Mitoses	Class
0	1002945	5	4	4	5	7	10	3	2	1	2
1	1015425	3	1	1	1	2	2	3	1	1	2
2	1016277	6	8	8	1	3	4	3	7	1	2
3	1017023	4	1	1	3	2	1	3	1	1	2
4	1017122	8	10	10	8	7	10	9	7	1	4

[8]:
```
df.drop('Id', axis=1, inplace=True)
```

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 698 entries, 0 to 697
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Clump_thickness              698 non-null    int64
 1   Uniformity_Cell_Size         698 non-null    int64
 2   Uniformity_Cell_Shape        698 non-null    int64
 3   Marginal_Adhesion            698 non-null    int64
 4   Single_Epithelial_Cell_Size  698 non-null    int64
 5   Bare_Nuclei                  698 non-null    object
 6   Bland_Chromatin              698 non-null    int64
 7   Normal_Nucleoli              698 non-null    int64
 8   Mitoses                      698 non-null    int64
```

```
 9   Class                        698 non-null    int64
dtypes: int64(9), object(1)
memory usage: 54.7+ KB
```

[11]:
```
for var in df.columns:
    print(df[var].value_counts())
```

```
Clump_thickness
1     145
5     129
3     108
4      80
10     69
2      50
8      46
6      34
7      23
9      14
Name: count, dtype: int64
Uniformity_Cell_Size
1     383
10     67
3      52
2      45
4      40
5      30
8      29
6      27
7      19
9       6
Name: count, dtype: int64
Uniformity_Cell_Shape
1     352
2      59
10     58
3      56
4      44
5      34
6      30
7      30
8      28
9       7
Name: count, dtype: int64
Marginal_Adhesion
1     406
3      58
2      58
10     55
```

```
4      33
8      25
5      23
6      22
7      13
9       5
Name: count, dtype: int64
Single_Epithelial_Cell_Size
2     385
3      72
4      48
1      47
6      41
5      39
10     31
8      21
7      12
9       2
Name: count, dtype: int64
Bare_Nuclei
1     401
10    132
2      30
5      30
3      28
8      21
4      19
?      16
9       9
7       8
6       4
Name: count, dtype: int64
Bland_Chromatin
2     166
3     164
1     152
7      73
4      40
5      34
8      28
10     20
9      11
6      10
Name: count, dtype: int64
Normal_Nucleoli
1     442
10     61
3      44
```

```
2      36
8      24
6      22
5      19
4      18
7      16
9      16
Name: count, dtype: int64
Mitoses
1     578
2      35
3      33
10     14
4      12
7       9
8       8
5       6
6       3
Name: count, dtype: int64
Class
2    457
4    241
Name: count, dtype: int64
```

[12]:
```
df['Bare_Nuclei'] = pd.to_numeric(df['Bare_Nuclei'], errors='coerce')
```

[13]:
```
df.dtypes
```

[13]:
```
Clump_thickness                int64
Uniformity_Cell_Size           int64
Uniformity_Cell_Shape          int64
Marginal_Adhesion              int64
Single_Epithelial_Cell_Size    int64
Bare_Nuclei                  float64
Bland_Chromatin                int64
Normal_Nucleoli                int64
Mitoses                        int64
Class                          int64
dtype: object
```

[14]:
```
df.isnull().sum()
```

[14]:
```
Clump_thickness                0
Uniformity_Cell_Size           0
Uniformity_Cell_Shape          0
Marginal_Adhesion              0
Single_Epithelial_Cell_Size    0
```

```
Bare_Nuclei                    16
Bland_Chromatin                 0
Normal_Nucleoli                 0
Mitoses                         0
Class                           0
dtype: int64
```

[15]:
```
df['Bare_Nuclei'].value_counts()
```

[15]:
```
Bare_Nuclei
1.0     401
10.0    132
2.0      30
5.0      30
3.0      28
8.0      21
4.0      19
9.0       9
7.0       8
6.0       4
Name: count, dtype: int64
```

[16]:
```
df['Bare_Nuclei'].unique()
```

[16]:
```
array([10.,  2.,  4.,  1.,  3.,  9.,  7., nan,  5.,  8.,  6.])
```

[17]:
```
df['Bare_Nuclei'].isna().sum()
```

[17]:
```
16
```

[18]:
```
df['Class'].value_counts()
```

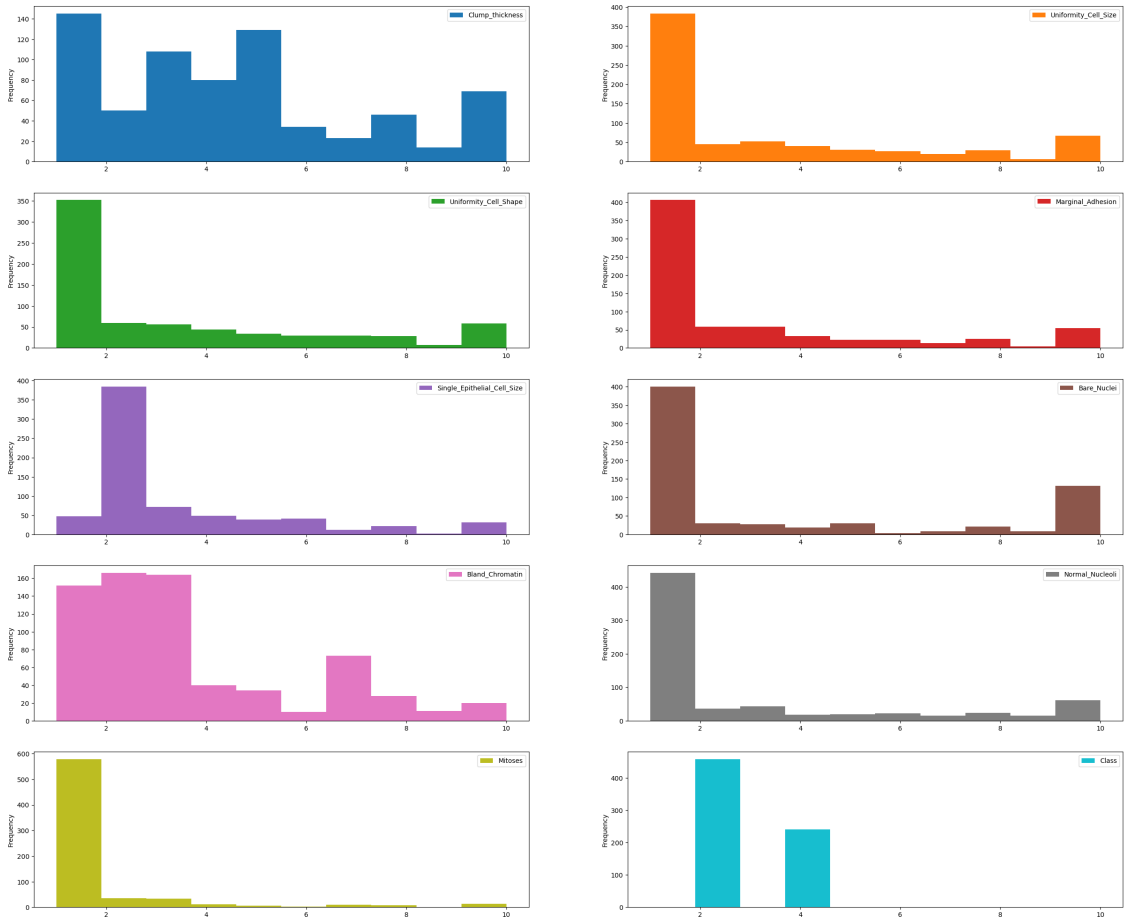
[18]:
```
Class
2    457
4    241
Name: count, dtype: int64
```

[20]:
```
print(round(df.describe(),2))
```

```
       Clump_thickness  Uniformity_Cell_Size  Uniformity_Cell_Shape  \
count           698.00                698.00                 698.00
mean              4.42                  3.14                   3.21
std               2.82                  3.05                   2.97
min               1.00                  1.00                   1.00
25%               2.00                  1.00                   1.00
50%               4.00                  1.00                   1.00
75%               6.00                  5.00                   5.00
```

```
max              10.00                 10.00                  10.00

       Marginal_Adhesion  Single_Epithelial_Cell_Size  Bare_Nuclei  \
count             698.00                       698.00       682.00
mean                2.81                         3.22         3.55
std                 2.86                         2.22         3.65
min                 1.00                         1.00         1.00
25%                 1.00                         2.00         1.00
50%                 1.00                         2.00         1.00
75%                 4.00                         4.00         6.00
max                10.00                        10.00        10.00

       Bland_Chromatin  Normal_Nucleoli  Mitoses   Class
count           698.00           698.00   698.00  698.00
mean              3.44             2.87     1.59    2.69
std               2.44             3.06     1.72    0.95
min               1.00             1.00     1.00    2.00
25%               2.00             1.00     1.00    2.00
50%               3.00             1.00     1.00    2.00
75%               5.00             4.00     1.00    4.00
max              10.00            10.00    10.00    4.00
```

[21]:
```
plt.rcParams['figure.figsize']=(30,25)

df.plot(kind='hist', bins=10, subplots=True, layout=(5,2), sharex=False,␣
 ↪sharey=False)

plt.show()
```



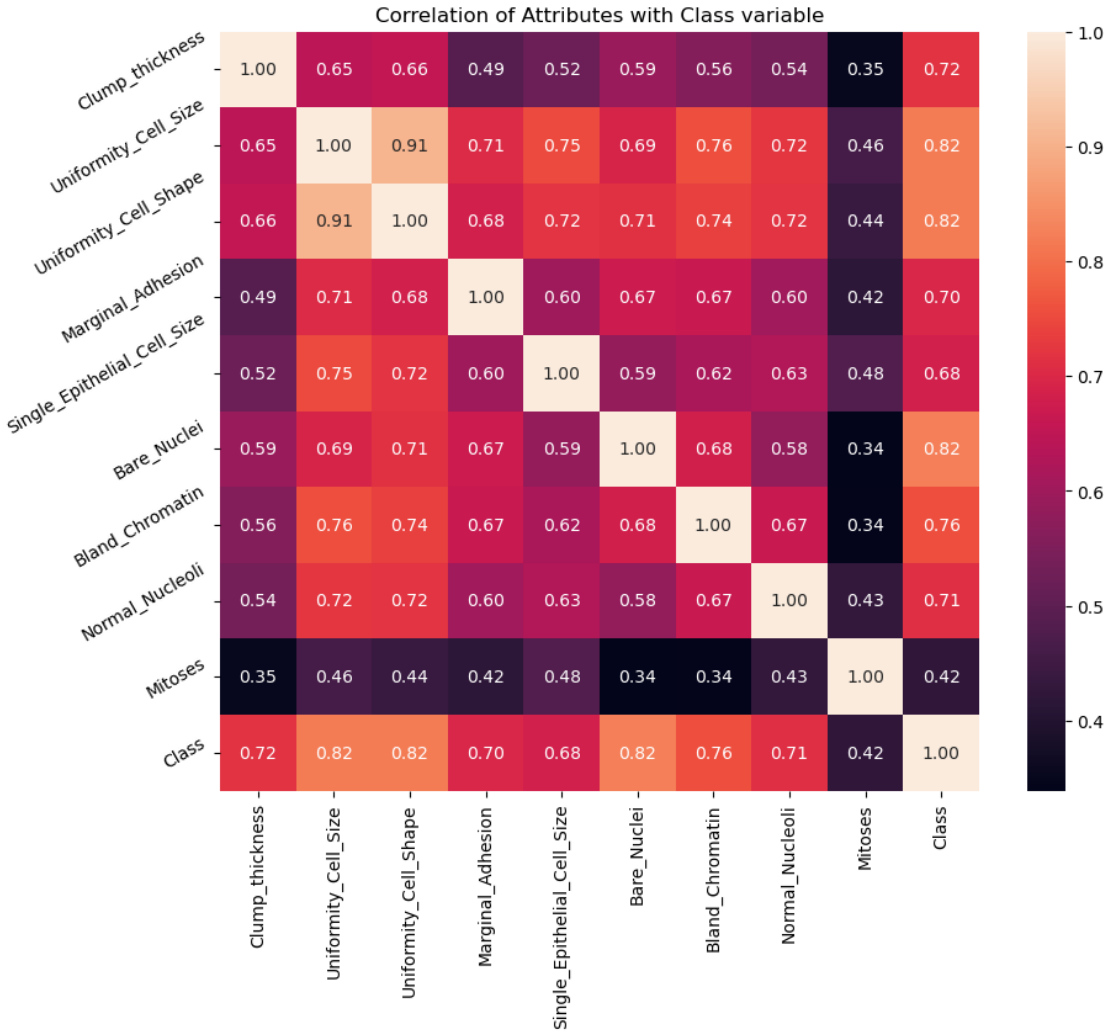
[22]:
```
correlation = df.corr()
```

[23]:
```
correlation['Class'].sort_values(ascending=False)
```

[23]:
```
Class                          1.000000
Bare_Nuclei                    0.822563
Uniformity_Cell_Shape          0.818794
Uniformity_Cell_Size           0.817772
Bland_Chromatin                0.756732
Clump_thickness                0.716509
Normal_Nucleoli                0.712067
Marginal_Adhesion              0.696605
Single_Epithelial_Cell_Size    0.682618
Mitoses                        0.423008
Name: Class, dtype: float64
```

[25]:
```
import seaborn as sns
```

[27]:
```
plt.figure(figsize=(10,8))
plt.title('Correlation of Attributes with Class variable')
a = sns.heatmap(correlation, square=True, annot=True, fmt='.2f',␣
 ↪linecolor='white')
a.set_xticklabels(a.get_xticklabels(), rotation=90)
a.set_yticklabels(a.get_yticklabels(), rotation=30)
plt.show()
```



[28]:
```
X = df.drop(['Class'], axis=1)

y = df['Class']
```

[29]:
```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
↪random_state = 0)
```

[30]:
```
X_train.shape, X_test.shape
```

[30]:
```
((558, 9), (140, 9))
```

[31]:
```
X_train.dtypes
```

[31]:
```
Clump_thickness                int64
Uniformity_Cell_Size           int64
Uniformity_Cell_Shape          int64
Marginal_Adhesion              int64
Single_Epithelial_Cell_Size    int64
Bare_Nuclei                  float64
Bland_Chromatin                int64
Normal_Nucleoli                int64
Mitoses                        int64
dtype: object
```

[32]:
```
X_train.isnull().sum()
```

[32]:
```
Clump_thickness                0
Uniformity_Cell_Size           0
Uniformity_Cell_Shape          0
Marginal_Adhesion              0
Single_Epithelial_Cell_Size    0
Bare_Nuclei                   15
Bland_Chromatin                0
Normal_Nucleoli                0
Mitoses                        0
dtype: int64
```

[33]:
```
X_test.isnull().sum()
```

[33]:
```
Clump_thickness                0
Uniformity_Cell_Size           0
Uniformity_Cell_Shape          0
Marginal_Adhesion              0
Single_Epithelial_Cell_Size    0
Bare_Nuclei                    1
Bland_Chromatin                0
Normal_Nucleoli                0
Mitoses                        0
dtype: int64
```

[34]:
```
for col in X_train.columns:
    if X_train[col].isnull().mean()>0:
        print(col, round(X_train[col].isnull().mean(),4))
```

```
Bare_Nuclei 0.0269
```

[35]:
```
X_train.isnull().sum()
```

[35]:
```
Clump_thickness               0
Uniformity_Cell_Size          0
Uniformity_Cell_Shape         0
Marginal_Adhesion             0
Single_Epithelial_Cell_Size   0
Bare_Nuclei                  15
Bland_Chromatin               0
Normal_Nucleoli               0
Mitoses                       0
dtype: int64
```

[36]:
```
for df1 in [X_train, X_test]:
    for col in X_train.columns:
        col_median=X_train[col].median()
        df1[col].fillna(col_median, inplace=True)
```

[37]:
```
X_train.isnull().sum()
```

[37]:
```
Clump_thickness               0
Uniformity_Cell_Size          0
Uniformity_Cell_Shape         0
Marginal_Adhesion             0
Single_Epithelial_Cell_Size   0
Bare_Nuclei                   0
Bland_Chromatin               0
Normal_Nucleoli               0
Mitoses                       0
dtype: int64
```

[38]:
```
X_test.isnull().sum()
```

[38]:
```
Clump_thickness               0
Uniformity_Cell_Size          0
Uniformity_Cell_Shape         0
Marginal_Adhesion             0
Single_Epithelial_Cell_Size   0
Bare_Nuclei                   0
Bland_Chromatin               0
Normal_Nucleoli               0
```

```
Mitoses                        0
dtype: int64
```

[39]:
```
X_train.head()
```

[39]:
```
     Clump_thickness  Uniformity_Cell_Size  Uniformity_Cell_Shape  \
62                 6                     3                      4
193                3                     1                      1
263                7                     9                      4
222                7                     5                      6
140                2                     1                      1

     Marginal_Adhesion  Single_Epithelial_Cell_Size  Bare_Nuclei  \
62                   1                            5          2.0
193                  1                            2          1.0
263                 10                           10          3.0
222                  3                            3          8.0
140                  1                            2          1.0

     Bland_Chromatin  Normal_Nucleoli  Mitoses
62                 3                9        1
193                3                1        1
263                5                3        3
222                7                4        1
140                1                1        1
```

[40]:
```
X_test.head()
```

[40]:
```
     Clump_thickness  Uniformity_Cell_Size  Uniformity_Cell_Shape  \
603                5                     3                      2
619                3                     1                      1
452                4                     5                      5
85                 3                     3                      6
416                1                     1                      1

     Marginal_Adhesion  Single_Epithelial_Cell_Size  Bare_Nuclei  \
603                  8                            5         10.0
619                  1                            2          1.0
452                  8                            6         10.0
85                   4                            5          8.0
416                  1                            2          1.0

     Bland_Chromatin  Normal_Nucleoli  Mitoses
603                8                1        2
619                2                1        1
452               10                7        1
85                 4                4        1
```

```
416                2                1         1
```

[41]:
```
cols = X_train.columns
```

[42]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

[43]:
```
X_train = pd.DataFrame(X_train, columns=[cols])
```

[44]:
```
X_test = pd.DataFrame(X_test, columns=[cols])
```

[45]:
```
X_train.head()
```

[45]:

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bland_Chromatin	Normal_Nucleoli	Mitoses
0	0.574621	-0.040143	0.277515	-0.629622	0.775913	-0.384119	-0.171342	1.983330	-0.333601
1	-0.497748	-0.680143	-0.721540	-0.629622	-0.549473	-0.661042	-0.171342	-0.601658	-0.333601
2	0.932077	1.879857	0.277515	2.541854	2.984890	-0.107196	0.660039	0.044589	0.859663
3	0.932077	0.599857	0.943552	0.075150	-0.107678	1.277420	1.491419	0.367712	-0.333601
4	-0.855205	-0.680143	-0.721540	-0.629622	-0.549473	-0.661042	-1.002722	-0.601658	-0.333601

[46]:
```
# import KNeighbors ClaSSifier from sklearn
from sklearn.neighbors import KNeighborsClassifier

# instantiate the model
knn = KNeighborsClassifier(n_neighbors=3)

# fit the model to the training set
knn.fit(X_train, y_train)
```

[46]:
```
KNeighborsClassifier(n_neighbors=3)
```

[47]:
```
y_pred = knn.predict(X_test)

y_pred
```

[47]:
```
array([4, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 2, 4,
       4, 4, 2, 4, 4, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 4,
       4, 4, 2, 4, 2, 4, 2, 2, 2, 4, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 2, 4,
       4, 2, 4, 4, 2, 2, 4, 2, 2, 2, 4, 2, 4, 2, 4, 2, 2, 2, 2, 2, 4, 2,
       2, 4, 4, 4, 2, 4, 2, 4, 2, 2, 2, 2, 4, 4, 4, 4, 2, 2, 4, 2, 2, 2,
       2, 4, 2, 2, 2, 2, 4, 2, 2, 4, 2, 2, 4, 4, 4, 2, 2, 4, 2, 2, 4, 4,
       2, 4, 2, 2, 2, 2, 4, 4], dtype=int64)
```

[48]:
```
knn.predict_proba(X_test)[:,0]
```

[48]:
```
array([0.        , 1.        , 0.        , 0.33333333, 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 0.        , 0.        ,
       1.        , 0.        , 0.        , 0.        , 1.        ,
       0.        , 0.        , 0.        , 0.66666667, 1.        ,
       0.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 0.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 0.        , 0.        ,
       0.        , 1.        , 0.        , 1.        , 0.        ,
       1.        , 1.        , 1.        , 0.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 0.        ,
       0.        , 0.33333333, 0.        , 0.        , 1.        ,
       0.        , 0.        , 1.        , 0.        , 0.        ,
       1.        , 1.        , 0.        , 1.        , 1.        ,
       1.        , 0.33333333, 1.        , 0.        , 1.        ,
       0.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 0.        , 1.        , 1.        , 0.        ,
       0.        , 0.        , 1.        , 0.33333333, 1.        ,
       0.        , 1.        , 1.        , 1.        , 1.        ,
       0.33333333, 0.        , 0.        , 0.        , 1.        ,
       1.        , 0.33333333, 1.        , 1.        , 1.        ,
       1.        , 0.33333333, 1.        , 0.66666667, 0.66666667,
       1.        , 0.        , 1.        , 1.        , 0.        ,
       1.        , 1.        , 0.        , 0.33333333, 0.        ,
       1.        , 1.        , 0.        , 1.        , 1.        ,
       0.        , 0.        , 1.        , 0.        , 1.        ,
       1.        , 1.        , 1.        , 0.        , 0.33333333])
```

[49]:
```
knn.predict_proba(X_test)[:,1]
```

[49]:
```
array([1.        , 0.        , 1.        , 0.66666667, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 1.        , 1.        ,
       0.        , 1.        , 1.        , 1.        , 0.        ,
       1.        , 1.        , 1.        , 0.33333333, 0.        ,
       1.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 1.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 1.        , 1.        ,
       1.        , 0.        , 1.        , 0.        , 1.        ,
       0.        , 0.        , 0.        , 1.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 1.        ,
       1.        , 0.66666667, 1.        , 1.        , 0.        ,
       1.        , 1.        , 0.        , 1.        , 1.        ,
       0.        , 0.        , 1.        , 0.        , 0.        ,
       0.        , 0.66666667, 0.        , 1.        , 0.        ,
       1.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 1.        , 0.        , 0.        , 1.        ,
       1.        , 1.        , 0.        , 0.66666667, 0.        ,
       1.        , 0.        , 0.        , 0.        , 0.        ,
       0.66666667, 1.        , 1.        , 1.        , 0.        ,
       0.        , 0.66666667, 0.        , 0.        , 0.        ,
       0.        , 0.66666667, 0.        , 0.33333333, 0.33333333,
       0.        , 1.        , 0.        , 0.        , 1.        ,
       0.        , 0.        , 1.        , 0.66666667, 1.        ,
       0.        , 0.        , 1.        , 0.        , 0.        ,
       1.        , 1.        , 0.        , 1.        , 0.        ,
       0.        , 0.        , 0.        , 1.        , 0.66666667])
```

[50]:
```
from sklearn.metrics import accuracy_score

print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score: 0.9714
```

[51]:
```
y_pred_train = knn.predict(X_train)
```

[52]:
```
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train,␣
↪y_pred_train)))
```

```
Training-set accuracy score: 0.9803
```

[53]:
```
print('Training set score: {:.4f}'.format(knn.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(knn.score(X_test, y_test)))
```

```
Training set score: 0.9803
Test set score: 0.9714
```

[54]:
```
y_test.value_counts()
```

[54]:
```
Class
2    85
4    55
Name: count, dtype: int64
```

[55]:
```
null_accuracy = (85/(85+55))

print('Null accuracy score: {0:0.4f}'. format(null_accuracy))
```

```
Null accuracy score: 0.6071
```

[56]:
```
knn_5 = KNeighborsClassifier(n_neighbors=5)

# fit the model to the training set
knn_5.fit(X_train, y_train)


# predict on the test-set
y_pred_5 = knn_5.predict(X_test)


print('Model accuracy score with k=5 : {0:0.4f}'. format(accuracy_score(y_test,␣
 ↪y_pred_5)))
```

```
Model accuracy score with k=5 : 0.9714
```

[57]:
```
knn_6 = KNeighborsClassifier(n_neighbors=6)


# fit the model to the training set
knn_6.fit(X_train, y_train)


# predict on the test-set
y_pred_6 = knn_6.predict(X_test)


print('Model accuracy score with k=6 : {0:0.4f}'. format(accuracy_score(y_test,␣
 ↪y_pred_6)))
```

```
Model accuracy score with k=6 : 0.9643
```

[58]:
```
knn_7 = KNeighborsClassifier(n_neighbors=7)


# fit the model to the training set
```

```
knn_7.fit(X_train, y_train)


# predict on the test-set
y_pred_7 = knn_7.predict(X_test)


print('Model accuracy score with k=7 : {0:0.4f}'. format(accuracy_score(y_test,␣
 ↪y_pred_7)))
```

```
Model accuracy score with k=7 : 0.9571
```

[59]:
```
knn_8 = KNeighborsClassifier(n_neighbors=8)


# fit the model to the training set
knn_8.fit(X_train, y_train)


# predict on the test-set
y_pred_8 = knn_8.predict(X_test)


print('Model accuracy score with k=8 : {0:0.4f}'. format(accuracy_score(y_test,␣
 ↪y_pred_8)))
```

```
Model accuracy score with k=8 : 0.9643
```

[60]:
```
knn_9 = KNeighborsClassifier(n_neighbors=9)


# fit the model to the training set
knn_9.fit(X_train, y_train)


# predict on the test-set
y_pred_9 = knn_9.predict(X_test)


print('Model accuracy score with k=9 : {0:0.4f}'. format(accuracy_score(y_test,␣
 ↪y_pred_9)))
```

```
Model accuracy score with k=9 : 0.9643
```

[61]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
```

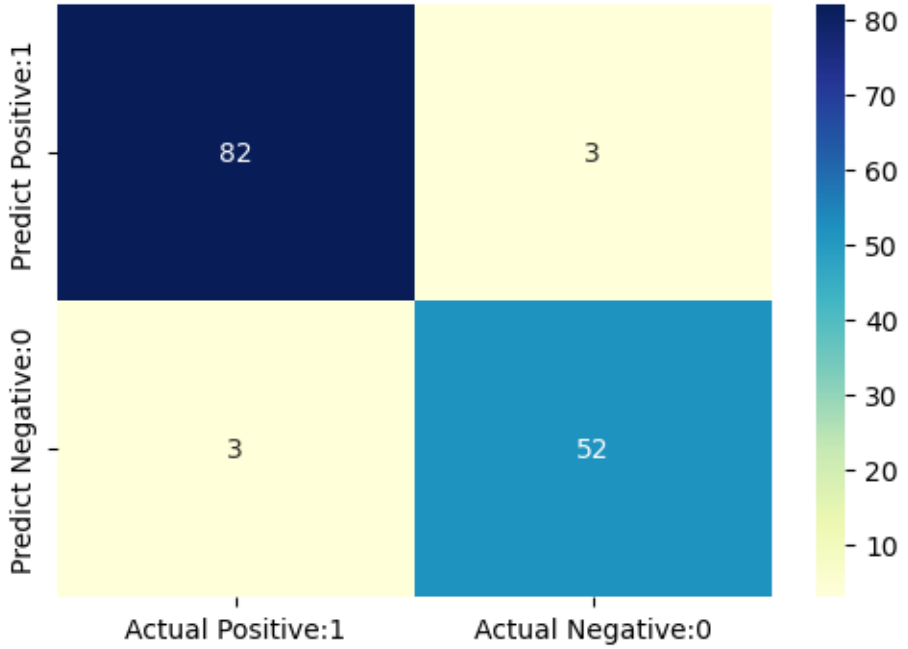
```
print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[83  2]
 [ 2 53]]

True Positives(TP) =  83

True Negatives(TN) =  53

False Positives(FP) =  2

False Negatives(FN) =  2
```

[62]:
```
cm_7 = confusion_matrix(y_test, y_pred_7)

print('Confusion matrix\n\n', cm_7)

print('\nTrue Positives(TP) = ', cm_7[0,0])

print('\nTrue Negatives(TN) = ', cm_7[1,1])

print('\nFalse Positives(FP) = ', cm_7[0,1])

print('\nFalse Negatives(FN) = ', cm_7[1,0])
```

```
Confusion matrix

 [[82  3]
 [ 3 52]]

True Positives(TP) =  82

True Negatives(TN) =  52

False Positives(FP) =  3

False Negatives(FN) =  3
```

[65]:
```
plt.figure(figsize=(6,4))

cm_matrix = pd.DataFrame(data=cm_7, columns=['Actual Positive:1', 'Actual␣
↪Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:
↪0'])

sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
plt.show()
```



[66]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred_7))
```

```
              precision    recall  f1-score   support

           2       0.96      0.96      0.96        85
           4       0.95      0.95      0.95        55

    accuracy                           0.96       140
   macro avg       0.96      0.96      0.96       140
weighted avg       0.96      0.96      0.96       140
```

[67]:
```
TP = cm_7[0,0]
TN = cm_7[1,1]
FP = cm_7[0,1]
FN = cm_7[1,0]
```

[68]:
```
classification_accuracy = (TP + TN) / float(TP + TN + FP + FN)

print('Classification accuracy : {0:0.4f}'.format(classification_accuracy))
```

```
Classification accuracy : 0.9571
```

[69]:
```
classification_error = (FP + FN) / float(TP + TN + FP + FN)

print('Classification error : {0:0.4f}'.format(classification_error))
```

```
Classification error : 0.0429
```

[70]:
```
precision = TP / float(TP + FP)


print('Precision : {0:0.4f}'.format(precision))
```

```
Precision : 0.9647
```

[71]:
```
recall = TP / float(TP + FN)

print('Recall or Sensitivity : {0:0.4f}'.format(recall))
```

```
Recall or Sensitivity : 0.9647
```

[72]:
```
true_positive_rate = TP / float(TP + FN)


print('True Positive Rate : {0:0.4f}'.format(true_positive_rate))
```

```
True Positive Rate : 0.9647
```

[73]:
```
false_positive_rate = FP / float(FP + TN)


print('False Positive Rate : {0:0.4f}'.format(false_positive_rate))
```

```
False Positive Rate : 0.0545
```

[74]:
```
specificity = TN / (TN + FP)

print('Specificity : {0:0.4f}'.format(specificity))
```

```
Specificity : 0.9455
```

[75]:
```
y_pred_prob = knn.predict_proba(X_test)[0:10]

y_pred_prob
```

[75]:
```
array([[0.        , 1.        ],
       [1.        , 0.        ],
       [0.        , 1.        ],
       [0.33333333, 0.66666667],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ]])
```

[76]:
```
y_pred_prob_df = pd.DataFrame(data=y_pred_prob, columns=['Prob of - benign␣
↪cancer (2)', 'Prob of - malignant cancer (4)'])

y_pred_prob_df
```

[76]:

	Prob of - benign cancer (2)	Prob of - malignant cancer (4)
0	0.000000	1.000000
1	1.000000	0.000000
2	0.000000	1.000000
3	0.333333	0.666667
4	1.000000	0.000000
5	1.000000	0.000000
6	1.000000	0.000000
7	1.000000	0.000000
8	1.000000	0.000000
9	1.000000	0.000000

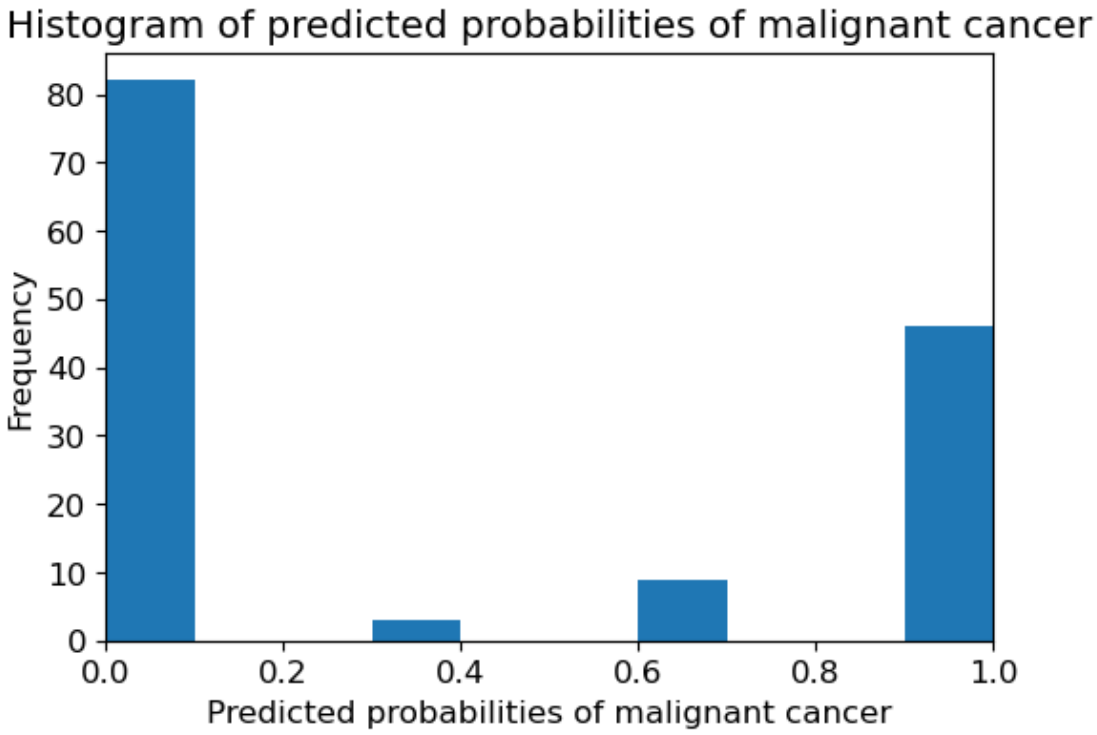
[77]:
```
knn.predict_proba(X_test)[0:10, 1]
```

[77]:
```
array([1.        , 0.        , 1.        , 0.66666667, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ])
```

[78]:
```
y_pred_1 = knn.predict_proba(X_test)[:, 1]
```

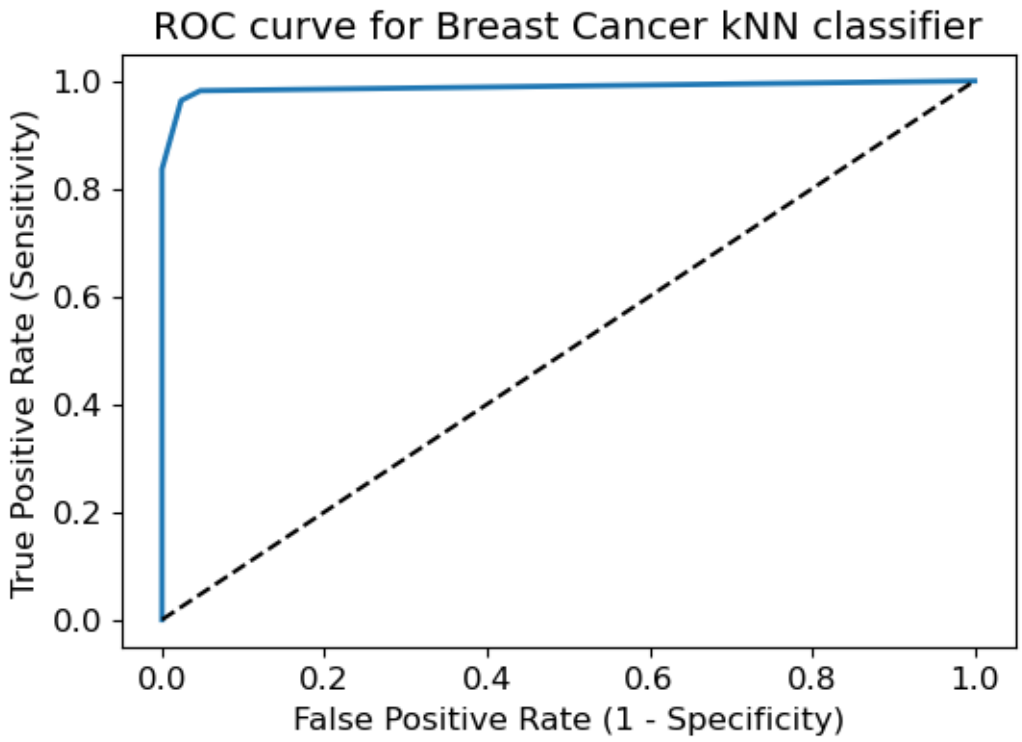
[81]:
```
plt.figure(figsize=(6,4))


# adjust the font size
plt.rcParams['font.size'] = 12
```

```
# plot histogram with 10 bins
plt.hist(y_pred_1, bins = 10)


# set the title of predicted probabilities
plt.title('Histogram of predicted probabilities of malignant cancer')


# set the x-axis limit
plt.xlim(0,1)


# set the title
plt.xlabel('Predicted probabilities of malignant cancer')
plt.ylabel('Frequency')

plt.show()
```



[82]:
```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, y_pred_1, pos_label=4)
```

```
plt.figure(figsize=(6,4))

plt.plot(fpr, tpr, linewidth=2)

plt.plot([0,1], [0,1], 'k--' )

plt.rcParams['font.size'] = 12

plt.title('ROC curve for Breast Cancer kNN classifier')

plt.xlabel('False Positive Rate (1 - Specificity)')

plt.ylabel('True Positive Rate (Sensitivity)')

plt.show()
```



[83]:
```
from sklearn.metrics import roc_auc_score

ROC_AUC = roc_auc_score(y_test, y_pred_1)

print('ROC AUC : {:.4f}'.format(ROC_AUC))
```

```
ROC AUC : 0.9883
```

[84]:
```
from sklearn.model_selection import cross_val_score

Cross_validated_ROC_AUC = cross_val_score(knn_7, X_train, y_train, cv=5,␣
 ↪scoring='roc_auc').mean()

print('Cross validated ROC AUC : {:.4f}'.format(Cross_validated_ROC_AUC))
```

```
Cross validated ROC AUC : 0.9811
```

[85]:
```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(knn_7, X_train, y_train, cv = 10, scoring='accuracy')

print('Cross-validation scores:{}'.format(scores))
```

```
Cross-validation scores:[0.96428571 0.98214286 0.96428571 0.98214286 0.96428571
0.94642857
 0.96428571 1.         0.98181818 0.96363636]
```

[86]:
```
print('Average cross-validation score: {:.4f}'.format(scores.mean()))
```

```
Average cross-validation score: 0.9713
```

2 Project Overview

This project applies the k-Nearest Neighbors (kNN) classification algorithm to predict whether a breast tumor is benign (2) or malignant (4) based on various cell attributes from the breast cancer dataset.

3 Steps Followed

3.1 1) Data Loading & Cleaning

- Dataset contained 698 rows and 11 columns.
- The Bare_Nuclei column had missing values (?) which were converted to numeric and imputed using the median.
- The Id column was dropped since it's not useful for prediction.

3.2 2) Exploratory Data Analysis (EDA)

- Distribution of features was analyzed using histograms.
- Correlation analysis revealed strong correlations of Class with Bare_Nuclei (0.82), Uniformity_Cell_Size (0.82), and Uniformity_Cell_Shape (0.82).

3.3 3) Data Preprocessing

- Dataset split into training (80%) and testing (20%) sets.

- Features standardized using StandardScaler to bring them to the same scale.

3.4 4) Model Training & Evaluation

- Trained kNN classifier with k=3,5,6,7,8,9.
- Best performance at k=3 & k=5, achieving 97.14% accuracy on the test set.
- Training accuracy: 98.03%, Test accuracy: 97.14%.
- Null accuracy (majority class baseline): 60.71%, showing strong improvement by the model.

3.5 5) Performance Metrics

- Confusion Matrix (k=3):
- True Positives (TP): 83
- True Negatives (TN): 53
- False Positives (FP): 2
- False Negatives (FN): 2
- Precision: 96.47%
- Recall/Sensitivity: 96.47%
- Specificity: 94.55%
- ROC AUC: 0.9883 (excellent).
- Cross-validated ROC AUC (k=7): 0.9811
- 10-fold CV accuracy: ~97.13%

4 Key Findings

- kNN performed extremely well for this dataset with accuracy above 97%.
- Model shows high recall, meaning it's effective in identifying malignant tumors (critical in medical diagnosis).
- The dataset was slightly imbalanced (benign: 457, malignant: 241), but the classifier still generalized well.

5 Conclusion

- The kNN classifier is a reliable model for predicting breast cancer diagnosis in this dataset. With proper preprocessing (handling missing values, scaling), it achieved high accuracy, precision, and recall, making it a good candidate for medical decision support.

[ ]: