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

In [2]:
```
import os
for dirname,_,filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname,filename))
```

In [3]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
df=pd.read_csv(r"D:\Data Science with AI\Data Science With AI\5th-september - KN
```

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

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

In [6]:
```
df.head()
```

Out[6]:

	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

In [7]:
```
col_names=['Id','Clump_thickness','Uniformity_Cell_Size','Uniformity_Cell_Shape'
           'Single_Epithelial_Cell_Size','Bare_Nuclei','Bland_Chromatin','Normal
df.columns=col_names
```

In [8]:
```
df.columns
```

Out[8]:
```
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')
```

In [9]:
```
df.head()
```

Out[9]:

	Id	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adh
0	1002945	5	4	4	
1	1015425	3	1	1	
2	1016277	6	8	8	
3	1017023	4	1	1	
4	1017122	8	10	10	

In [10]:
```
df.drop('Id',axis=1,inplace=True)
```

In [11]:
```
df.head()
```

Out[11]:

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion	Sin
0	5	4	4	5	
1	3	1	1	1	
2	6	8	8	1	
3	4	1	1	3	
4	8	10	10	8	

In [12]:
```
df.info
```

Out[12]:

```
<bound method DataFrame.info of      Clump_thickness  Uniformity_Cell_Size  Uni
formity_Cell_Shape  \
0                  5                     4                      4
1                  3                     1                      1
2                  6                     8                      8
3                  4                     1                      1
4                  8                    10                     10
..               ...                   ...                    ...
693                3                     1                      1
694                2                     1                      1
695                5                    10                     10
696                4                     8                      6
697                4                     8                      8

     Marginal_Adhesion  Single_Epithelial_Cell_Size Bare_Nuclei  \
0                    5                            7          10
1                    1                            2           2
2                    1                            3           4
3                    3                            2           1
4                    8                            7          10
..                 ...                          ...         ...
693                  1                            3           2
694                  1                            2           1
695                  3                            7           3
696                  4                            3           4
697                  5                            4           5

     Bland_Chromatin  Normal_Nucleoli  Mitoses  Class
0                  3                2        1      2
1                  3                1        1      2
2                  3                7        1      2
3                  3                1        1      2
4                  9                7        1      4
..               ...              ...      ...    ...
693                1                1        1      2
694                1                1        1      2
695                8               10        2      4
696               10                6        1      4
697               10                4        1      4

[698 rows x 10 columns]>
```

In [13]:

```
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
```

In [14]:

```
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
7      30
6      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
```

In [15]:
```
df['Bare_Nuclei']=pd.to_numeric(df['Bare_Nuclei'],errors='coerce')
```

In [16]:
```
df.dtypes
```

```
Out[16]: 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
```

```
In [17]: df.isnull().sum()
```

```
Out[17]: 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
```

```
In [18]: df.isna().sum()
```

```
Out[18]: 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
```

```
In [19]: df['Bare_Nuclei'].value_counts()
```

```
Out[19]: 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
```

```
In [20]: df['Bare_Nuclei'].unique()
```

```
Out[20]: array([10.,  2.,  4.,  1.,  3.,  9.,  7., nan,  5.,  8.,  6.])
```

```
In [21]: df['Bare_Nuclei'].isna().sum()
```

```
Out[21]: np.int64(16)
```

```
In [22]: df['Class'].value_counts()
```

```
Out[22]: Class
         2    457
         4    241
         Name: count, dtype: int64
```

```
In [23]: df['Class'].value_counts()
```

```
Out[23]: Class
         2    457
         4    241
         Name: count, dtype: int64
```

```
In [24]: df['Class'].value_counts()
```

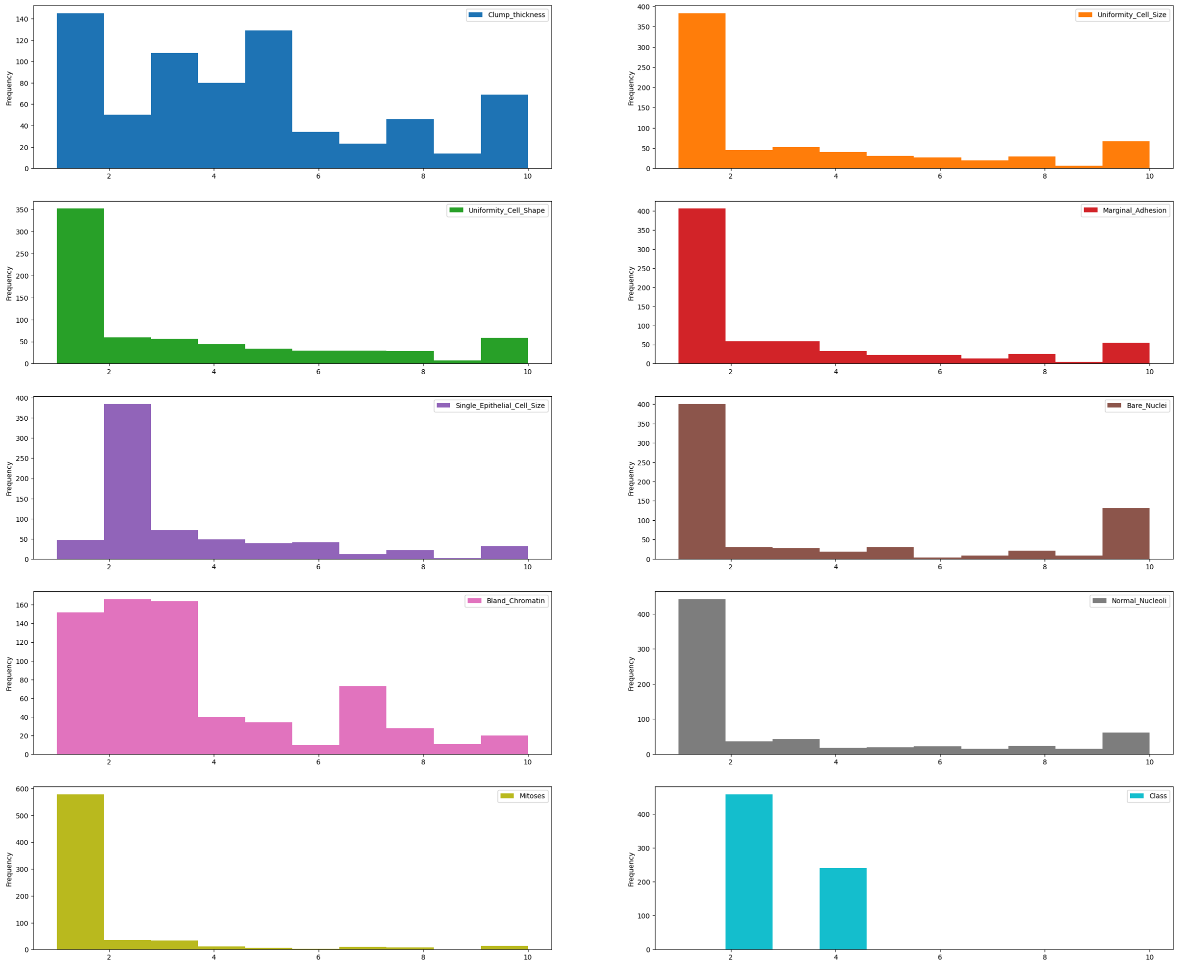
```
Out[24]: Class
         2    457
         4    241
         Name: count, dtype: int64
```

```
In [25]: print(round(df.describe(),2))
```

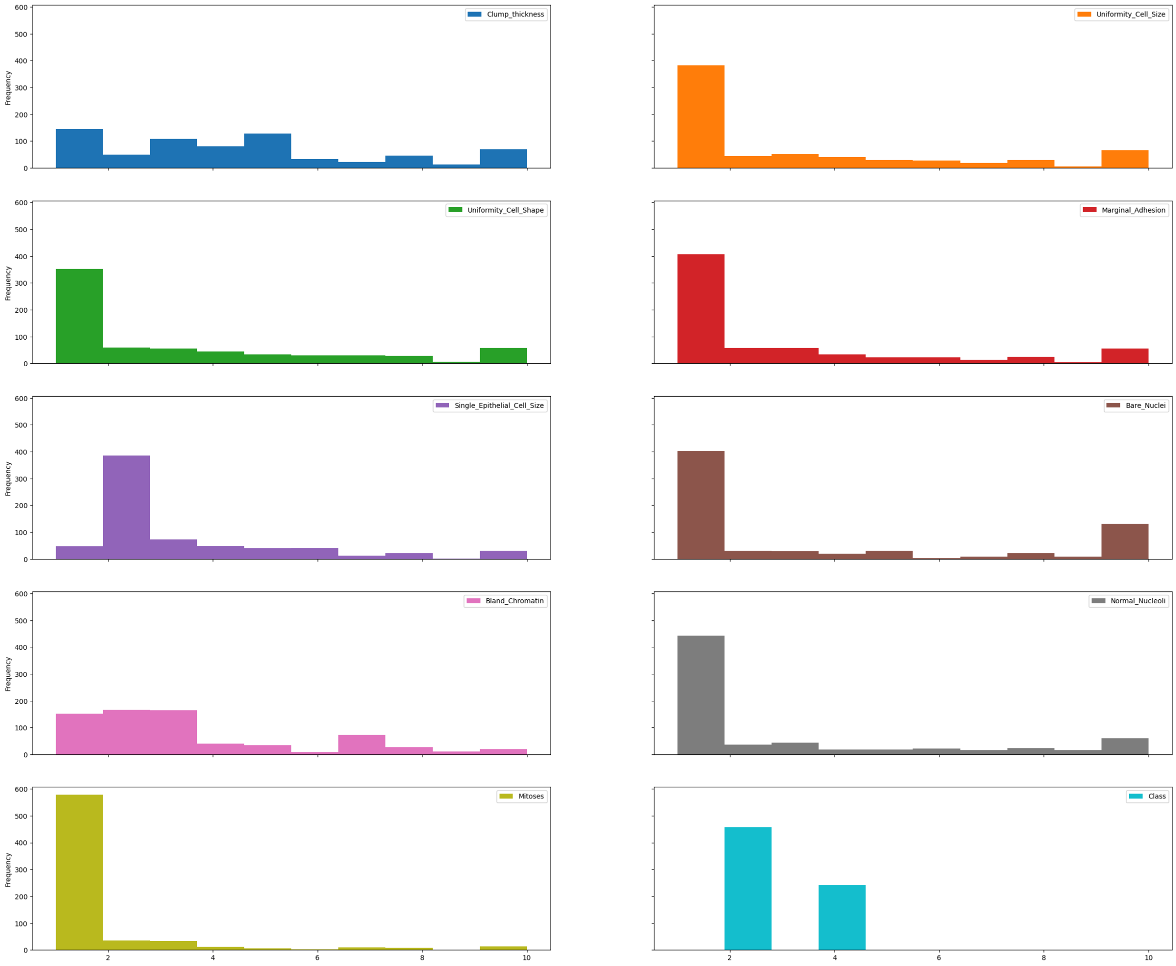
```
       Clump_thickness  Uniformity_Cell_Size  Uniformity_Cell_Shape  \
count           698.00                698.00                 698.00   
mean              4.42                  3.14                   3.21   
std               2.82                  3.05                   2.97   
min               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  
```

Data Visualization

In [27]:
```
plt.rcParams['figure.figsize']=(30,25)
df.plot(kind='hist',bins=10,subplots=True,layout=(5,2),sharex=False,sharey=False
plt.show()
```



In [29]:
```
plt.rcParams['figure.figsize']=(30,25)
df.plot(kind='hist',bins=10,subplots=True,layout=(5,2),sharex=True,sharey=True)
plt.show()
```



In [30]:

```
correlation=df.corr()
```

In [31]:

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

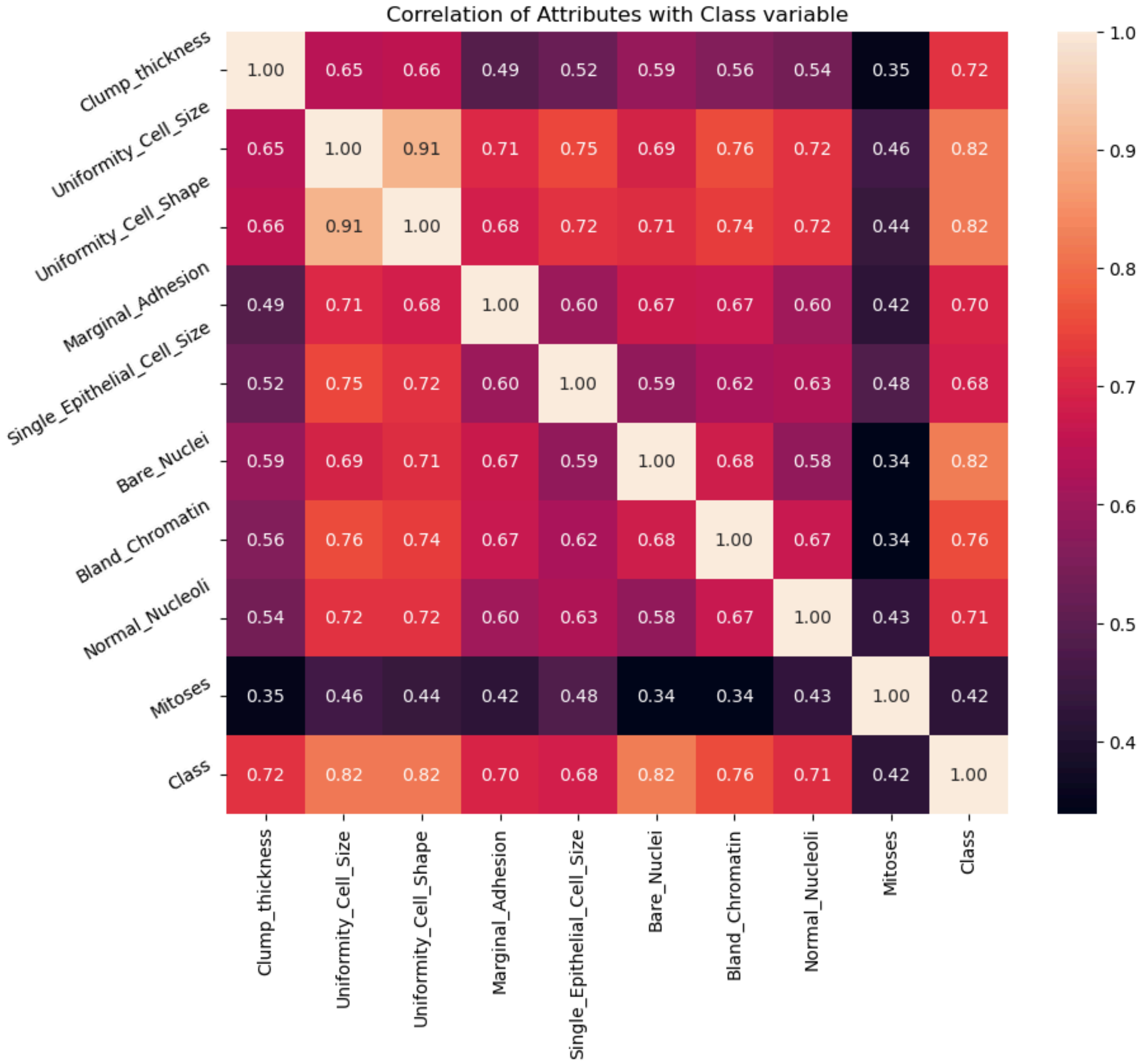
Out[31]:

```
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
```

Correlation Heat Map

In [32]:

```
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()
```



In [33]:
```
x=df.drop(['Class'],axis=1)
y=df['Class']
```

In [34]:
```
x
```

Out[34]:

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion
0	5	4	4	5
1	3	1	1	1
2	6	8	8	1
3	4	1	1	3
4	8	10	10	8
...	...	...	...	...
693	3	1	1	1
694	2	1	1	1
695	5	10	10	3
696	4	8	6	4
697	4	8	8	5

698 rows × 9 columns

In [35]:

```
y
```

Out[35]:

```
0      2
1      2
2      2
3      2
4      4
      ..
693    2
694    2
695    4
696    4
697    4
Name: Class, Length: 698, dtype: int64
```

10.split data into seperate training and test set

In [37]:

```
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)
```

In [38]:

```
x_train.shape,x_test.shape
```

Out[38]:

```
((558, 9), (140, 9))
```

In [39]:

```
x_train
```

Out[39]:

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion
62	6	3	4	1
193	3	1	1	1
263	7	9	4	10
222	7	5	6	3
140	2	1	1	1
...	...	...	...	...
359	6	10	10	10
192	1	1	1	1
629	6	2	3	1
559	5	1	1	1
684	1	1	1	1

558 rows × 9 columns

In [40]:
```
x_test
```

Out[40]:

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion
603	5	3	2	8
619	3	1	1	1
452	4	5	5	8
85	3	3	6	4
416	1	1	1	1
...	...	...	...	...
643	2	1	1	1
534	1	1	3	2
249	1	2	2	1
45	3	7	7	4
283	7	4	5	10

140 rows × 9 columns

In [41]:
```
x_train.dtypes
```

```
Out[41]: 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
```

```
In [42]: x_test.dtypes
```

```
Out[42]: 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
```

```
In [43]: y_train
```

```
Out[43]: 62     4
         193    2
         263    4
         222    4
         140    2
               ..
         359    4
         192    2
         629    2
         559    2
         684    2
         Name: Class, Length: 558, dtype: int64
```

```
In [44]: x_train.isnull().sum()
```

```
Out[44]: 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
```

```
In [45]: x_test.isnull().sum()
```

Out[45]:

```
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
```

In [46]:

```
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
```

In [47]:

```
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)
```

In [48]:

```
x_train.isnull().sum()
```

Out[48]:

```
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
```

In [49]:

```
x_test.isnull().sum()
```

Out[49]:

```
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
```

In [50]:

```
x_train.head()
```

Out[50]:

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion
62	6	3	4	1
193	3	1	1	1
263	7	9	4	10
222	7	5	6	3
140	2	1	1	1

In [51]:

```
x_test.head()
```

Out[51]:

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion
603	5	3	2	8
619	3	1	1	1
452	4	5	5	8
85	3	3	6	4
416	1	1	1	1

In [52]:

```
cols=x_train.columns
```

In [53]:

```
cols
```

Out[53]:

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

In [54]:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

In [55]:

```
x_train
```

Out[55]:

```
array([[ 0.5746208 , -0.04014337,  0.27751542, ..., -0.17134185,
         1.98333024, -0.33360075],
       [-0.49774845, -0.68014338, -0.72154008, ..., -0.17134185,
        -0.60165843, -0.33360075],
       [ 0.93207722,  1.87985665,  0.27751542, ...,  0.66003861,
         0.04458874,  0.85966347],
       ...,
       [ 0.5746208 , -0.36014337, -0.05550308, ..., -1.00272232,
        -0.60165843, -0.33360075],
       [ 0.21716438, -0.68014338, -0.72154008, ..., -0.17134185,
        -0.60165843, -0.33360075],
       [-1.21266128, -0.68014338, -0.72154008, ..., -1.00272232,
        -0.60165843, -0.33360075]])
```

In [56]:

```
x_test
```

Out[56]:

```
array([[ 0.21716438, -0.04014337, -0.38852158, ...,  1.90710931,
        -0.60165843,  0.26303136],
       [-0.49774845, -0.68014338, -0.72154008, ..., -0.58703209,
        -0.60165843, -0.33360075],
       [-0.14029203,  0.59985664,  0.61053392, ...,  2.73848978,
         1.33708307, -0.33360075],
       ...,
       [-1.21266128, -0.36014337, -0.38852158, ..., -1.00272232,
        -0.60165843, -0.33360075],
       [-0.49774845,  1.23985664,  1.27657092, ...,  0.24434838,
         1.66020665, -0.33360075],
       [ 0.93207722,  0.27985663,  0.61053392, ..., -0.17134185,
         1.66020665,  0.26303136]])
```

In [62]:

```
x_train=pd.DataFrame(x_train,columns=[cols])
```

In [63]:

```
x_train
```

Out[63]:

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion
0	0.574621	-0.040143	0.277515	-0.629622
1	-0.497748	-0.680143	-0.721540	-0.629622
2	0.932077	1.879857	0.277515	2.541854
3	0.932077	0.599857	0.943552	0.075150
4	-0.855205	-0.680143	-0.721540	-0.629622
...	...	...	...	...
553	0.574621	2.199857	2.275626	2.541854
554	-1.212661	-0.680143	-0.721540	-0.629622
555	0.574621	-0.360143	-0.055503	-0.629622
556	0.217164	-0.680143	-0.721540	-0.629622
557	-1.212661	-0.680143	-0.721540	-0.629622

558 rows × 9 columns

In [64]:

```
x_test=pd.DataFrame(x_test,columns=[cols])
```

In [65]:

```
x_test
```

Out[65]:

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion
0	0.217164	-0.040143	-0.388522	1.837081
1	-0.497748	-0.680143	-0.721540	-0.629622
2	-0.140292	0.599857	0.610534	1.837081
3	-0.497748	-0.040143	0.943552	0.427537
4	-1.212661	-0.680143	-0.721540	-0.629622
...	...	...	...	...
135	-0.855205	-0.680143	-0.721540	-0.629622
136	-1.212661	-0.680143	-0.055503	-0.277236
137	-1.212661	-0.360143	-0.388522	-0.629622
138	-0.497748	1.239857	1.276571	0.427537
139	0.932077	0.279857	0.610534	2.541854

140 rows × 9 columns

In [66]:

```
x_train.head()
```

Out[66]:

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion
0	0.574621	-0.040143	0.277515	-0.629622
1	-0.497748	-0.680143	-0.721540	-0.629622
2	0.932077	1.879857	0.277515	2.541854
3	0.932077	0.599857	0.943552	0.075150
4	-0.855205	-0.680143	-0.721540	-0.629622

In [67]:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)
```

Out[67]:

```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [68]:

```
knn
```

Out[68]:

```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [69]:

```
y_pred=knn.predict(x_test)
y_pred
```

```
Out[69]: 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])
```

```
In [70]: knn.predict_proba(x_test)[:,0]
```

```
Out[70]: 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])
```

```
In [71]: knn.predict_proba(x_test)[:,1]
```

Out[71]:

```
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])
```

In [72]:

```python
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
```

In [73]:

```python
from sklearn.metrics import accuracy_score
print('Model accuracy score:{0:0.4f}'.format(accuracy_score(y_train,y_pred)))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[73], line 2
      1 from sklearn.metrics import accuracy_score
----> 2 print('Model accuracy score:{0:0.4f}'.format(accuracy_score(y_train,y_pred)))

File D:\New folder\Lib\site-packages\sklearn\utils\_param_validation.py:216, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    210 try:
    211     with config_context(
    212         skip_parameter_validation=(
    213             prefer_skip_nested_validation or global_skip_validation
    214         )
    215     ):
--> 216         return func(*args, **kwargs)
    217 except InvalidParameterError as e:
    218     # When the function is just a wrapper around an estimator, we allow
    219     # the function to delegate validation to the estimator, but we replace
    220     # the name of the estimator by the name of the function in the error
    221     # message to avoid confusion.
    222     msg = re.sub(
    223         r"parameter of \w+ must be",
    224         f"parameter of {func.__qualname__} must be",
    225         str(e),
    226     )

File D:\New folder\Lib\site-packages\sklearn\metrics\_classification.py:227, in accuracy_score(y_true, y_pred, normalize, sample_weight)
    225 # Compute accuracy for each possible representation
    226 y_true, y_pred = attach_unique(y_true, y_pred)
--> 227 y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    228 check_consistent_length(y_true, y_pred, sample_weight)
    230 if y_type.startswith("multilabel"):

File D:\New folder\Lib\site-packages\sklearn\metrics\_classification.py:98, in _check_targets(y_true, y_pred)
     71 """Check that y_true and y_pred belong to the same classification task.
     72 
     73 This converts multiclass or binary types to a common shape, and raises a
   (...)
     95 y_pred : array or indicator matrix
     96 """
     97 xp, _ = get_namespace(y_true, y_pred)
---> 98 check_consistent_length(y_true, y_pred)
     99 type_true = type_of_target(y_true, input_name="y_true")
    100 type_pred = type_of_target(y_pred, input_name="y_pred")

File D:\New folder\Lib\site-packages\sklearn\utils\validation.py:475, in check_consistent_length(*arrays)
    473 uniques = np.unique(lengths)
    474 if len(uniques) > 1:
--> 475     raise ValueError(
    476         "Found input variables with inconsistent numbers of samples: %r"
    477         % [int(l) for l in lengths]
    478     )

ValueError: Found input variables with inconsistent numbers of samples: [558, 140]
```

In [74]:

```
y_pred_train=knn.predict(x_train)
```

In [75]:

```
y_pred_train
```

Out[75]:

```
array([4, 2, 4, 4, 2, 2, 2, 4, 2, 2, 2, 4, 2, 4, 2, 2, 2, 2, 2, 4, 4, 2,
       2, 2, 4, 2, 2, 2, 4, 2, 2, 2, 2, 4, 4, 2, 4, 2, 2, 2, 2, 2, 4, 2,
       4, 2, 2, 4, 4, 2, 2, 4, 2, 2, 2, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2,
       4, 4, 2, 2, 2, 2, 2, 2, 4, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 4, 4, 2,
       2, 2, 2, 2, 2, 2, 4, 2, 2, 4, 2, 4, 4, 4, 4, 2, 4, 2, 2, 2, 4, 4,
       2, 4, 2, 2, 2, 4, 4, 2, 2, 2, 2, 4, 4, 2, 2, 2, 2, 4, 2, 2, 2, 2,
       2, 2, 4, 4, 2, 4, 4, 4, 4, 2, 4, 2, 4, 4, 4, 2, 2, 4, 2, 2, 2, 4,
       2, 2, 2, 4, 4, 2, 4, 2, 2, 4, 4, 2, 4, 2, 4, 4, 2, 2, 4, 2, 4, 2,
       4, 2, 4, 2, 4, 2, 2, 2, 2, 4, 2, 2, 2, 2, 4, 2, 2, 4, 2, 2, 4, 2,
       4, 2, 4, 4, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 2, 4, 2, 2,
       2, 2, 4, 2, 4, 2, 4, 2, 2, 4, 4, 4, 2, 4, 4, 2, 2, 2, 2, 2, 4, 2,
       2, 2, 4, 2, 4, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 4, 2, 2, 4, 2, 4,
       4, 4, 4, 2, 4, 2, 2, 2, 4, 2, 4, 2, 2, 2, 4, 2, 4, 2, 4, 4, 4, 2,
       4, 2, 2, 2, 4, 2, 2, 2, 4, 4, 2, 4, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4,
       2, 2, 4, 4, 4, 2, 2, 2, 2, 2, 4, 4, 4, 2, 2, 4, 2, 4, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 2, 2, 2, 4, 2, 4, 2, 2, 2, 4, 2, 2,
       2, 4, 2, 2, 2, 2, 2, 2, 4, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4,
       2, 2, 2, 4, 2, 2, 2, 4, 4, 2, 2, 2, 4, 2, 4, 2, 2, 4, 4, 2, 2, 2,
       2, 4, 4, 2, 2, 2, 4, 2, 2, 4, 2, 2, 4, 2, 4, 2, 2, 4, 2, 4, 2, 2,
       2, 2, 2, 4, 2, 2, 2, 2, 2, 4, 4, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 4, 2, 2, 4, 2, 4, 4, 2, 4, 4, 4, 2, 4, 4, 2, 4, 2, 4,
       2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 4, 2, 4, 2, 2, 2, 2, 2, 2, 4, 2,
       2, 2, 2, 2, 2, 4, 2, 2, 2, 4, 2, 2, 4, 2, 2, 2, 2, 2, 4, 4, 2, 2,
       2, 2, 4, 2, 2, 4, 2, 2, 2, 4, 4, 2, 2, 4, 2, 2, 4, 2, 2, 2, 2, 4,
       4, 2, 2, 2, 2, 4, 4, 2, 2, 4, 2, 2, 2, 4, 2, 2, 4, 2, 2, 2, 2, 4,
       2, 2, 2, 4, 2, 2, 2, 2])
```

In [76]:

```
print('Training-set accuracy score:{0:0.4f}'.format(accuracy_score(y_train,y_pre
```

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

In [77]:

```
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
```

In [78]:

```
y_test.value_counts()
```

Out[78]:

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

In [80]:

```
null_accuracy=(85/(85+55))
print('Null accuracy score:{0:0.4f}'.format(null_accuracy))
```

```
Null accuracy score:0.6071
```

In [82]:

```
knn_5=KNeighborsClassifier(n_neighbors=5)
knn_5.fit(x_train,y_train)
y_pred_5=knn_5.predict(x_test)
print('Model accuracy score with k=5:{0:0.4f}'.format(accuracy_score(y_test,y_pr
```

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

In [83]:

```
knn_6=KNeighborsClassifier(n_neighbors=6)
knn_6.fit(x_train,y_train)
y_pred_6=knn_6.predict(x_test)
print('Model accuracy score with k=6:{0:0.4f}'.format(accuracy_score(y_test,y_pr
```

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

In [84]:

```
knn_7=KNeighborsClassifier(n_neighbors=7)
knn_7.fit(x_train,y_train)
y_pred_7=knn_7.predict(x_test)
print('Model accuracy score with k=7:{0:0.4f}'.format(accuracy_score(y_test,y_pr
```

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

In [85]:

```
knn_8=KNeighborsClassifier(n_neighbors=8)
knn_8.fit(x_train,y_train)
y_pred_8=knn_8.predict(x_test)
print('Model accuracy score with k=8:{0:0.4f}'.format(accuracy_score(y_test,y_pr
```

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

In [86]:

```
knn_9=KNeighborsClassifier(n_neighbors=9)
knn_9.fit(x_train,y_train)
y_pred_9=knn_9.predict(x_test)
print('Model accuracy score with k=9:{0:0.4f}'.format(accuracy_score(y_test,y_pr
```

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

In [88]:

```
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 Negative(TN)=',cm[1,1])
print('\nFalse Positive(FP)=',cm[0,1])
print('\nFalse Negative(FN)=',cm[1,0])
```

```
Confusion matrix

 [[83  2]
 [ 2 53]]

True Positives(TP)= 83

True Negative(TN)= 53

False Positive(FP)= 2

False Negative(FN)= 2
```

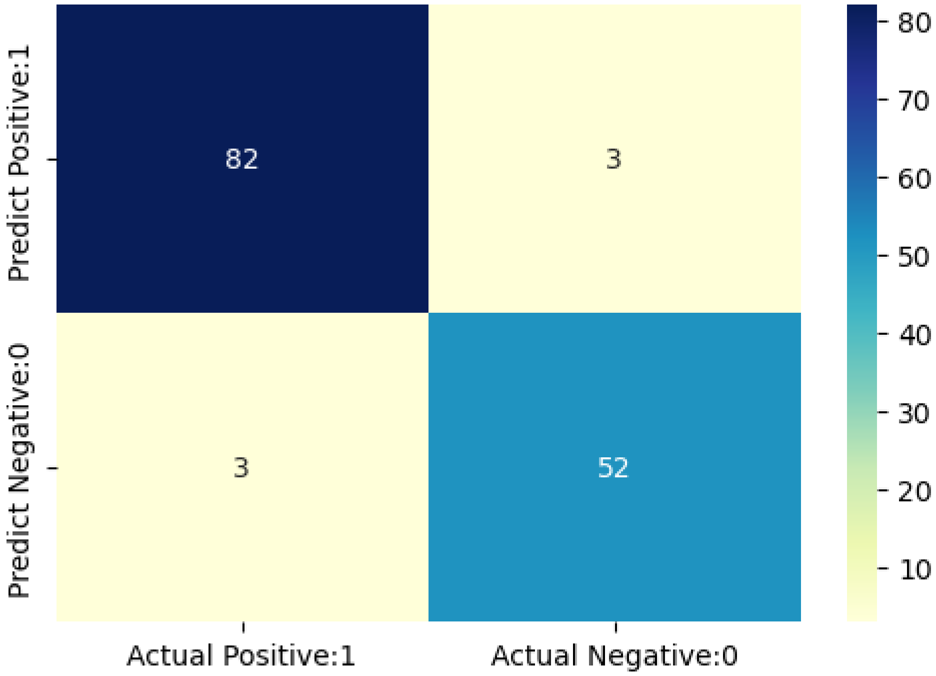
In [89]:

```
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 Negative(TN)=',cm_7[1,1])
print('\nFalse Positive(FP)=',cm_7[0,1])
print('\nFalse Negative(FN)=',cm_7[1,0])
```

```
confusion matrix

 [[82  3]
 [ 3 52]]

True Positives(TP)= 82

True Negative(TN)= 52

False Positive(FP)= 3

False Negative(FN)= 3
```

In [91]:
```
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')
```

Out[91]:
```
<Axes: >
```



In [92]:
```
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
```

In [93]:
```
TP=cm_7[0,0]
TN=cm_7[1,1]
FP=cm_7[0,1]
FN=cm_7[1,0]
```

In [94]:
```
classification_accuracy=(TP+TN)/float(TP+TN+FP+FN)
print('Classification accuracy:{0:0.4f}'.format(classification_accuracy))
```

```
Classification accuracy:0.9571
```

In [95]:
```
classification_error=(FP+FN)/float(TP+TN+FP+FN)
print('Classification error:{0:0.4f}'.format(classification_error))
```

```
Classification error:0.0429
```

In [96]:
```
precision=TP/float(TP+FP)
print('Precision:{0:0.4f}'.format(precision))
```

```
Precision:0.9647
```

In [97]:
```
recall=TP/float(TP+FN)
print('Recall or Sensitivity:{0:0.4f}'.format(recall))
```

```
Recall or Sensitivity:0.9647
```

In [98]:
```
true_positive_rate=TP/float(TP+FN)
print('True Positive Rate:{0:0.4f}'.format(true_positive_rate))
```

```
True Positive Rate:0.9647
```

In [99]:
```
false_positive_rate=FP/float(FP+TN)
print('False Positive Rate:{0:0.4f}'.format(false_positive_rate))
```

```
False Positive Rate:0.0545
```

In [100…
```
specificity=TN/(TN+FP)
print('Specificity:{0:0.4f}'.format(specificity))
```

```
Specificity:0.9455
```

In [101…
```
y_pred_prob=knn.predict_proba(x_test)[0:10]
y_pred_prob
```

Out[101…
```
array([[0.        , 1.        ],
       [1.        , 0.        ],
       [0.        , 1.        ],
       [0.33333333, 0.66666667],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ]])
```

In [102…
```
y_pred_prob_df=pd.DataFrame(data=y_pred_prob,columns=['Prob of - benign cancer (
y_pred_prob_df
```

Out[102…

	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

In [103…

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

Out[103…

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

In [104…

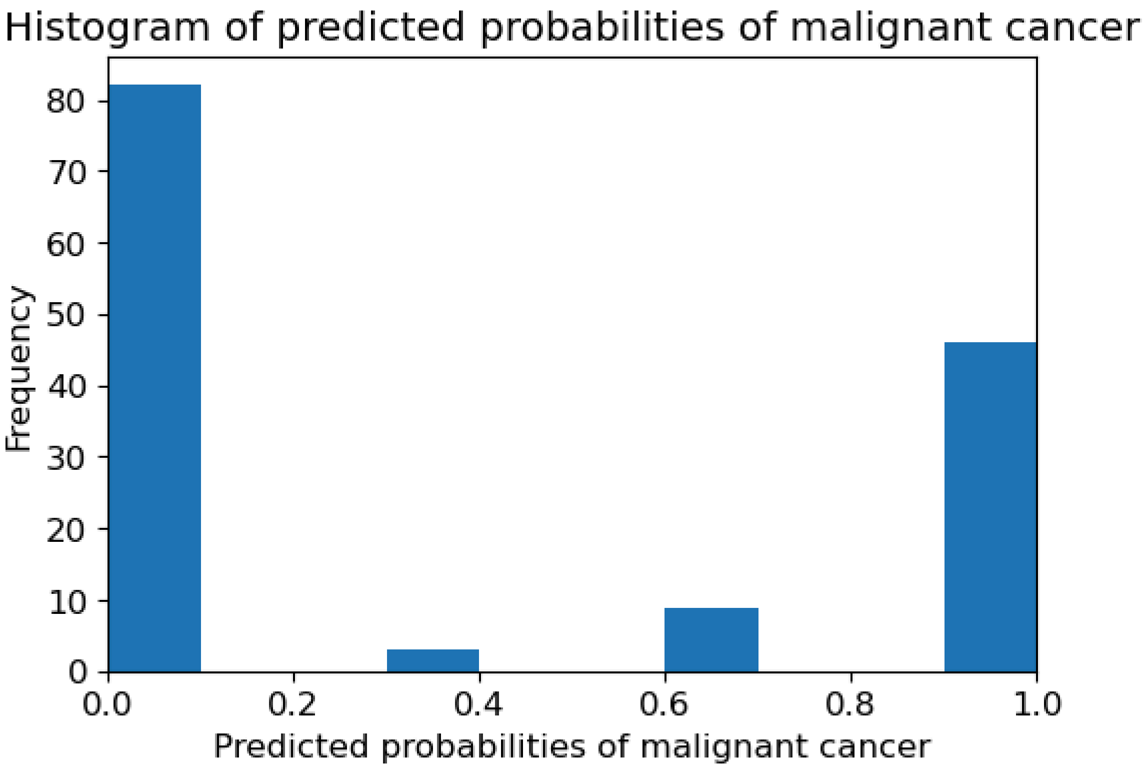
```
y_pred_1=knn.predict_proba(x_test)[:,1]
```

In [105…

```
plt.figure(figsize=(6,4))
plt.rcParams['font.size']=12
plt.hist(y_pred_1,bins=10)
plt.title('Histogram of predicted probabilities of malignant cancer')
plt.xlim(0,1)
plt.xlabel('Predicted probabilities of malignant cancer')
plt.ylabel('Frequency')
```

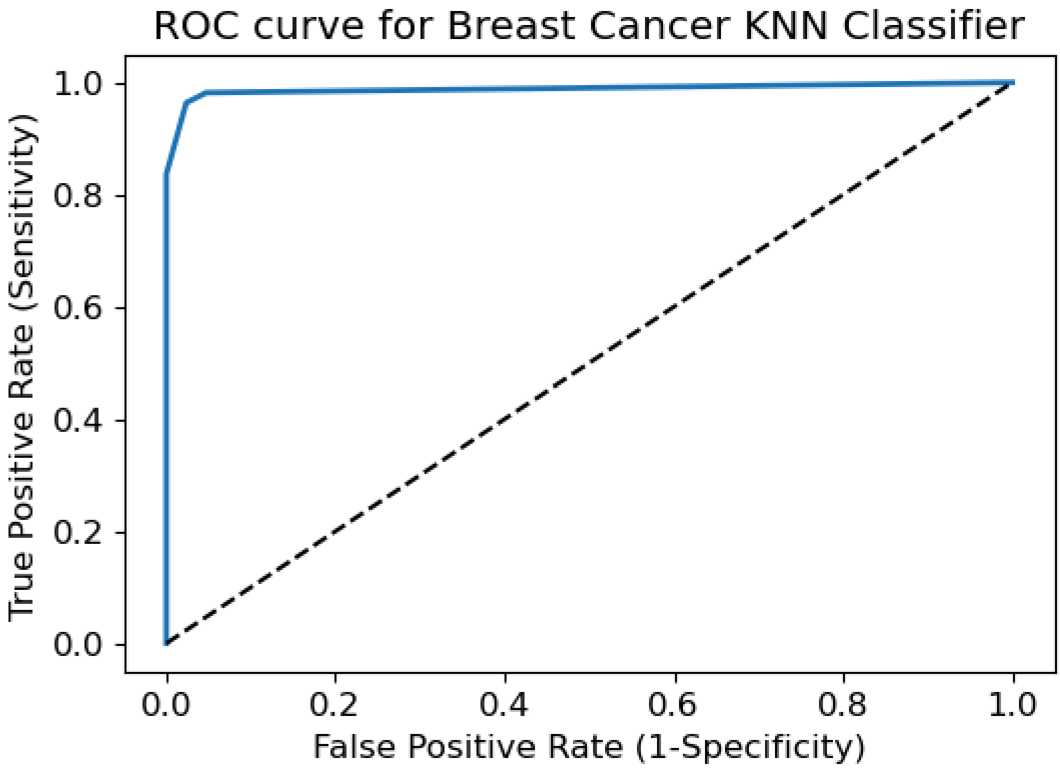
Out[105…

```
Text(0, 0.5, 'Frequency')
```



In [106…

```
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()
```



In [107…

```
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
```

In [108…

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

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

In [109…

```
from sklearn.model_selection import cross_val_score
scores=cross_val_score(knn_7,x_trian,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]
```

In [110…

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

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

In [ ]: