In [2]:

```python
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```python
df=pd.read_csv(r'C:\Users\sharma17\Downloads\ml\classification dataset\Breast_Cancer.csv')
```

In [4]:

```python
df.head()
```

Out[4]:

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

In [5]:

```python
df.drop(['Sample code number'],inplace=True,axis=1)
```

In [6]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683 entries, 0 to 682
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Clump Thickness              683 non-null    int64
 1   Uniformity of Cell Size      683 non-null    int64
 2   Uniformity of Cell Shape     683 non-null    int64
 3   Marginal Adhesion            683 non-null    int64
 4   Single Epithelial Cell Size  683 non-null    int64
 5   Bare Nuclei                  683 non-null    int64
 6   Bland Chromatin              683 non-null    int64
 7   Normal Nucleoli              683 non-null    int64
 8   Mitoses                      683 non-null    int64
 9   Class                        683 non-null    int64
dtypes: int64(10)
memory usage: 53.5 KB
```

In [7]:

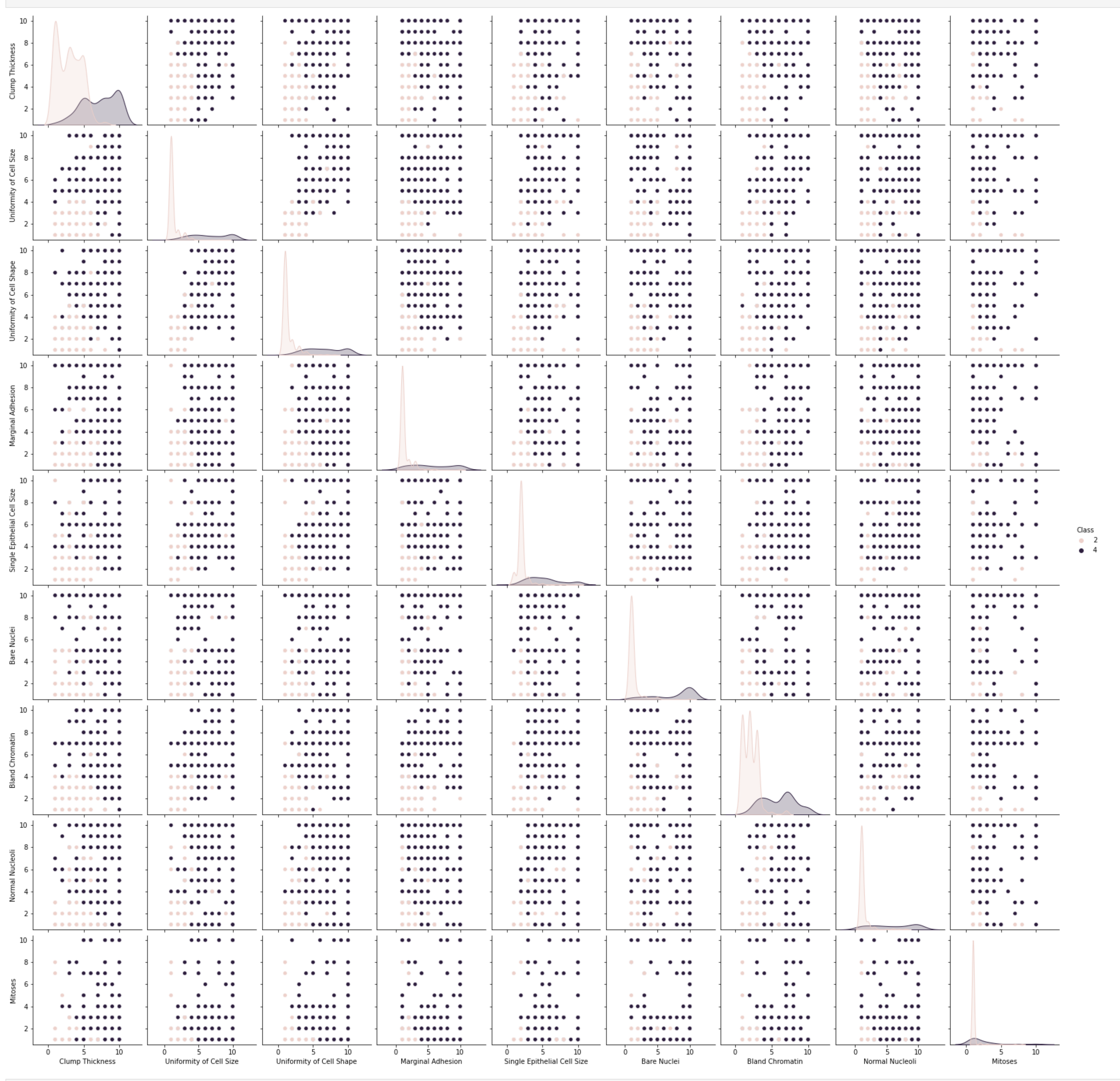
```python
df.describe()
```

Out[7]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
count	683.000000	683.000000	683.000000	683.000000	683.000000	683.000000	683.000000	683.000000	683.000000	683.000000
mean	4.442167	3.150805	3.215227	2.830161	3.234261	3.544656	3.445095	2.869693	1.603221	2.699854
std	2.820761	3.065145	2.988581	2.864562	2.223085	3.643857	2.449697	3.052666	1.732674	0.954592
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	2.000000
25%	2.000000	1.000000	1.000000	1.000000	2.000000	1.000000	2.000000	1.000000	1.000000	2.000000
50%	4.000000	1.000000	1.000000	1.000000	2.000000	1.000000	3.000000	1.000000	1.000000	2.000000
75%	6.000000	5.000000	5.000000	4.000000	4.000000	6.000000	5.000000	4.000000	1.000000	4.000000
max	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	4.000000

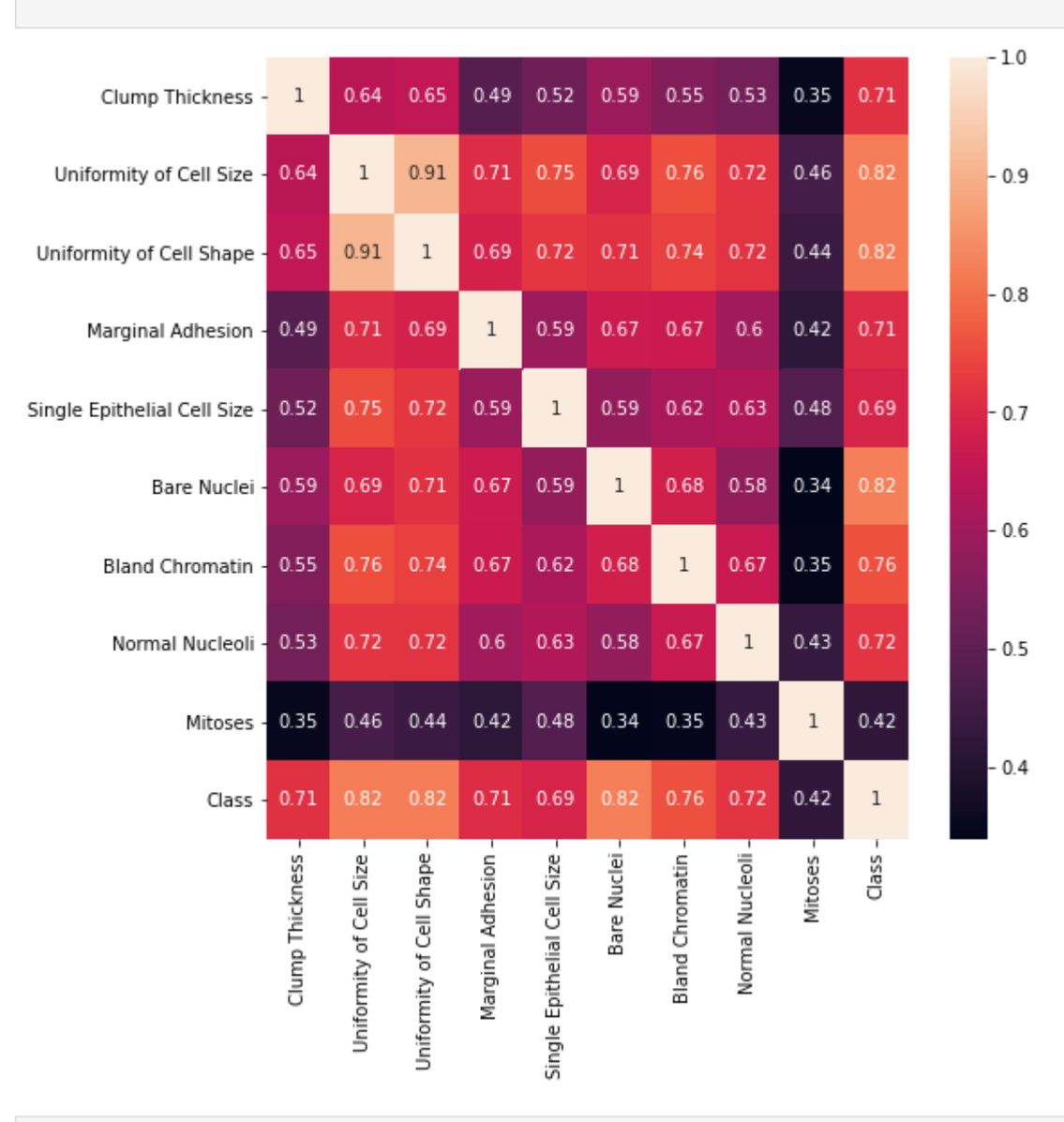
In [12]:

```python
sns.pairplot(df,hue='Class')
plt.show()
```



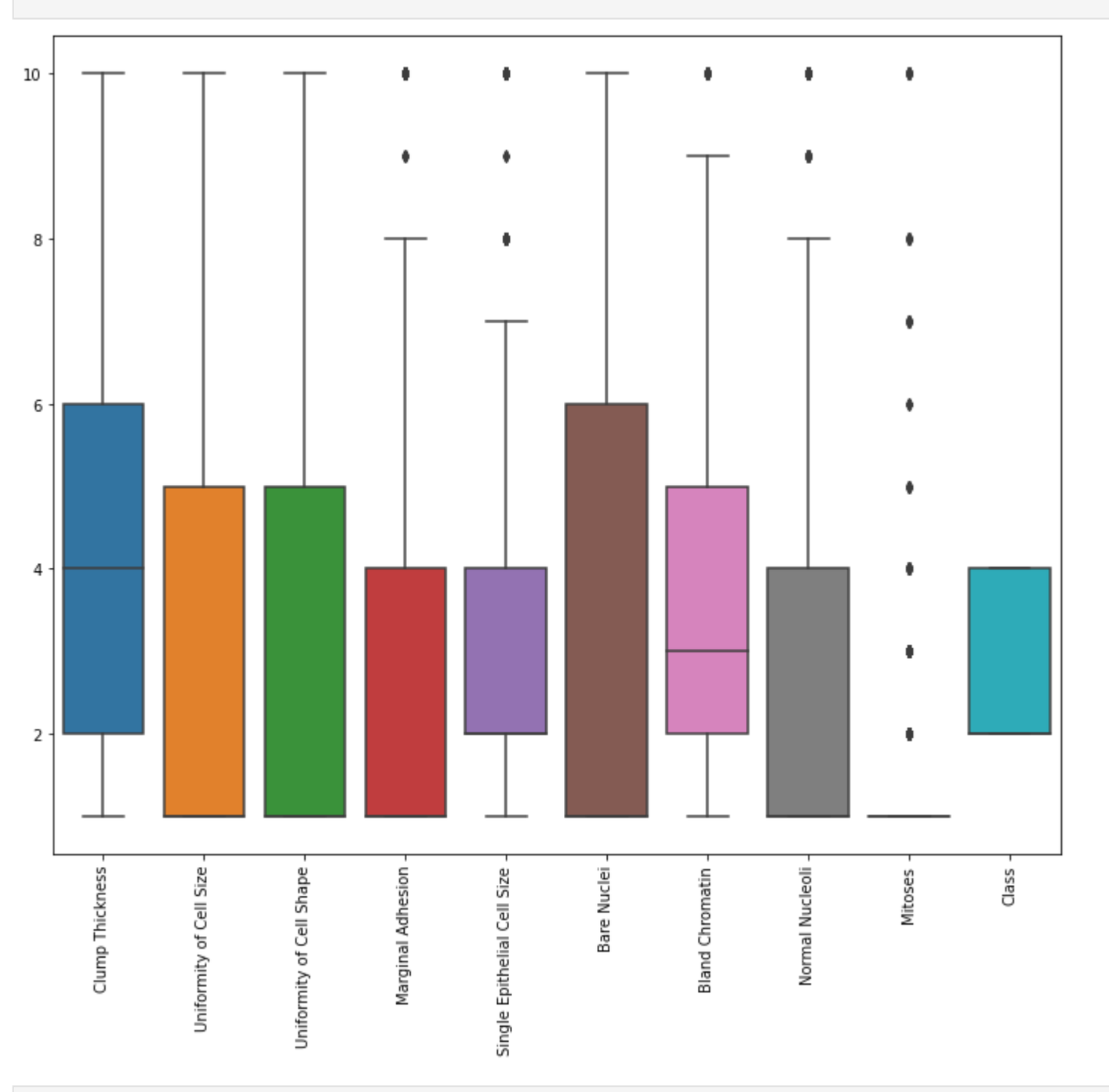
In [15]:

```python
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [34]:

```python
plt.figure(figsize=(12,10))
sns.boxplot(data=df)
plt.xticks(rotation=90)
plt.show()
```



In [42]:

```python
df['Single Epithelial Cell Size'].value_counts()
```

Out[42]:

```
2     376
3      71
4      48
1      44
6      40
5      39
10     31
8      21
7      11
9       2
Name: Single Epithelial Cell Size, dtype: int64
```

In [44]:

```python
df['Mitoses'].value_counts()
```

Out[44]:

```
1     563
2      35
3      33
10     14
4      12
7       9
8       8
5       6
6       3
Name: Mitoses, dtype: int64
```

In [54]:

```python
df.skew()
```

Out[54]:

```
Clump Thickness                0.587654
Uniformity of Cell Size        1.226404
Uniformity of Cell Shape       1.157890
Marginal Adhesion              1.509181
Single Epithelial Cell Size    1.703716
Bare Nuclei                    0.990016
Bland Chromatin                1.095270
Normal Nucleoli                1.420431
Mitoses                        2.917222
Class                          0.630694
dtype: float64
```

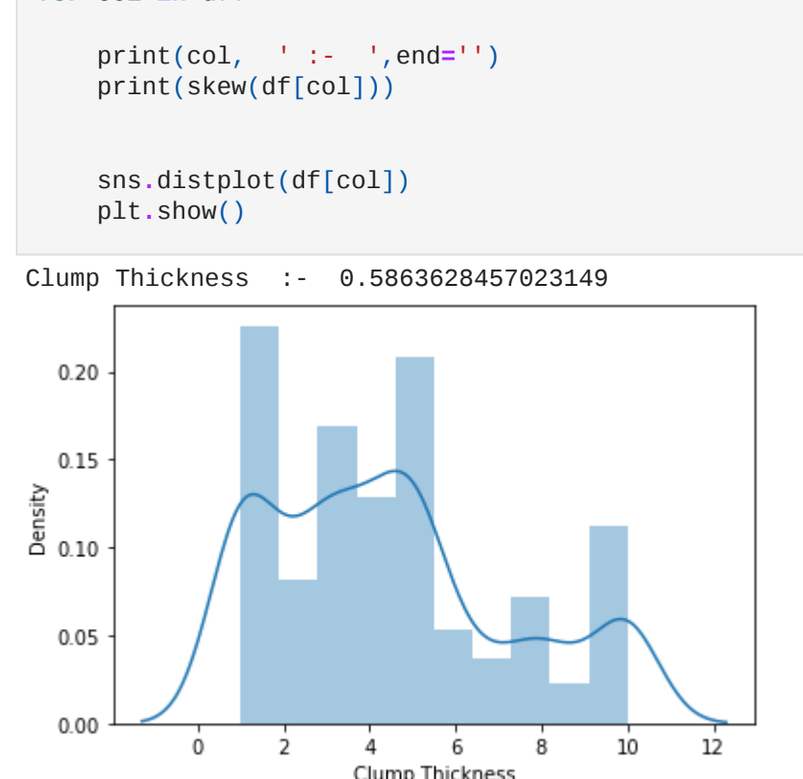
In [55]:

```python
from scipy.stats import skew
```

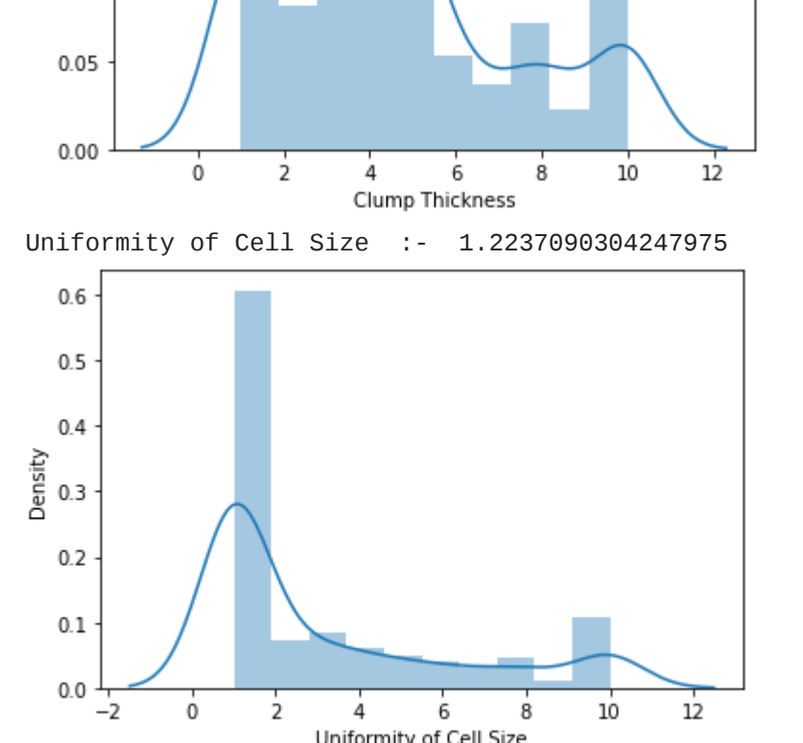
In [56]:

```python
for col in df:

    print(col,  ' :-  ',end='')
    print(skew(df[col]))


    sns.distplot(df[col])
    plt.show()
```

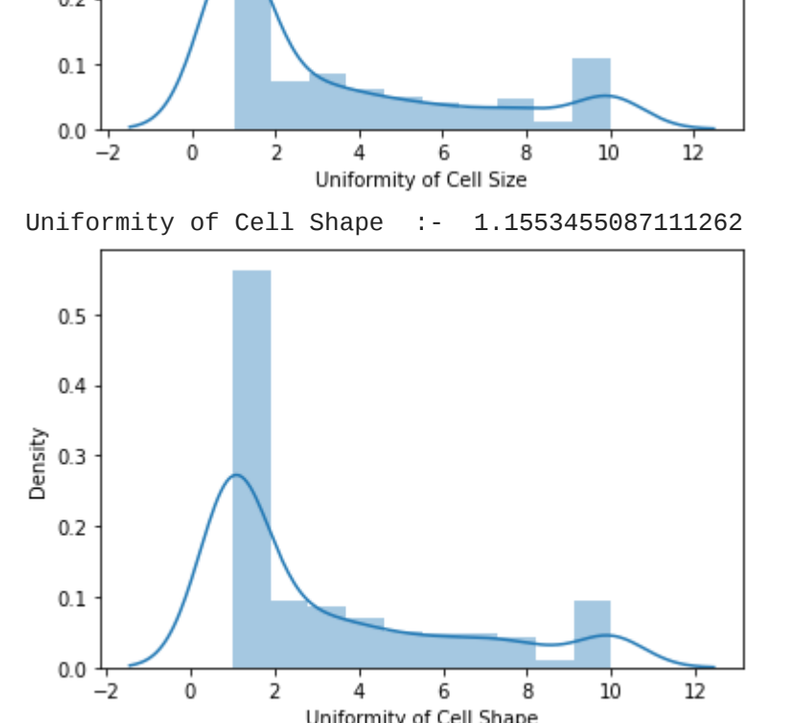
```
Clump Thickness  :-  0.5863628457023149
```



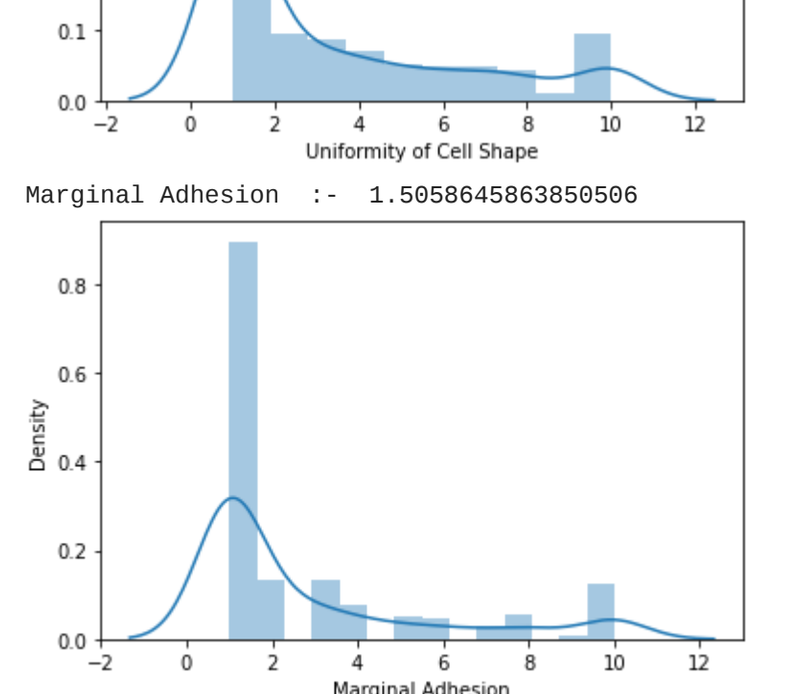

```
Uniformity of Cell Size  :-  1.2237090304247975
```



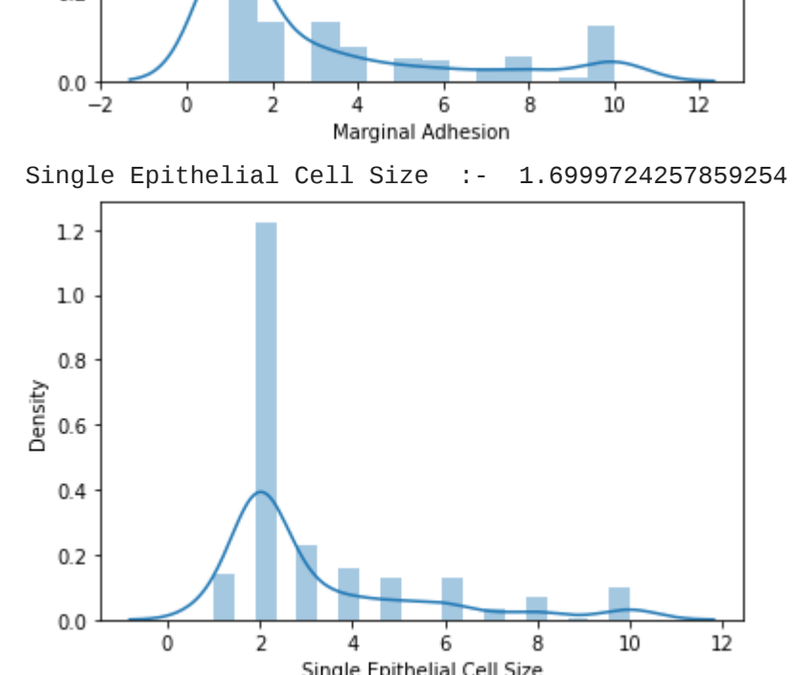

```
Uniformity of Cell Shape  :-  1.1553455087111262
```



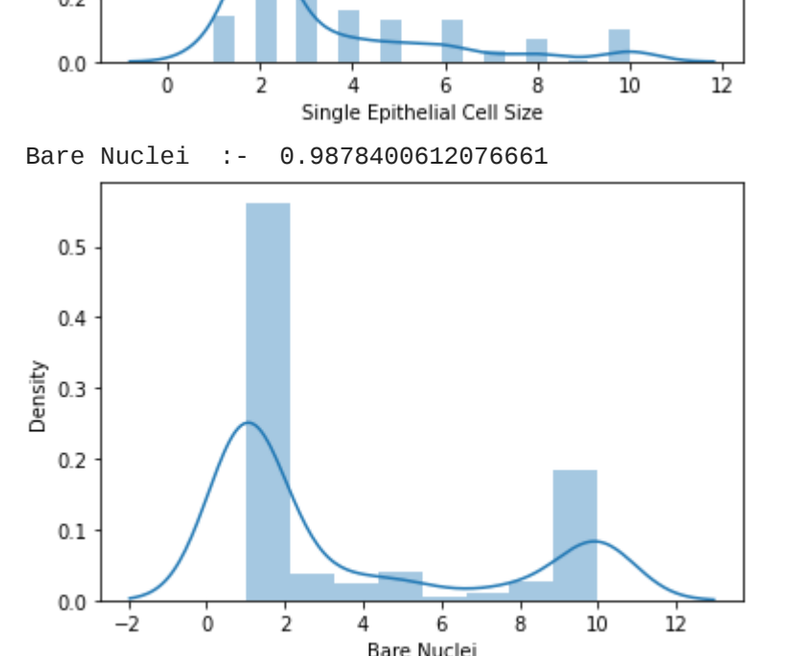

```
Marginal Adhesion  :-  1.5058645863850506
```



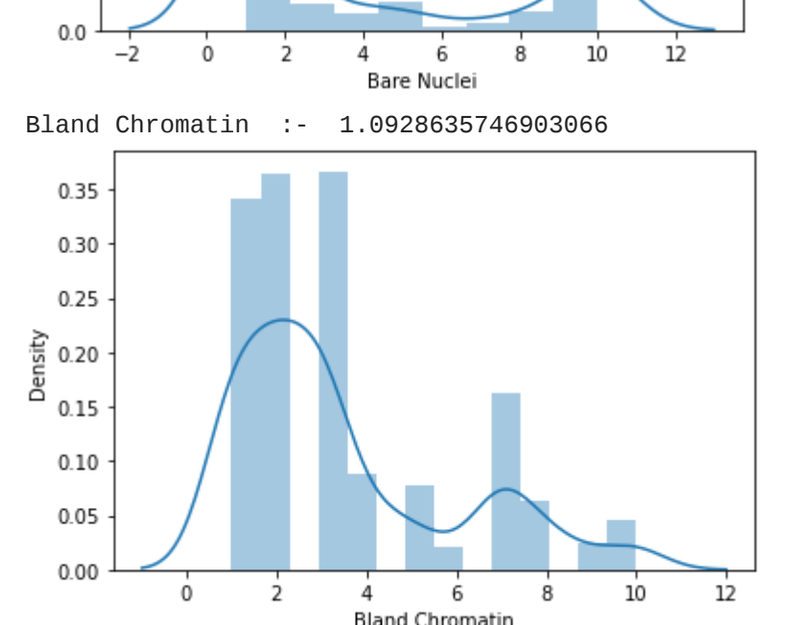

```
Single Epithelial Cell Size  :-  1.6999724257859254
```



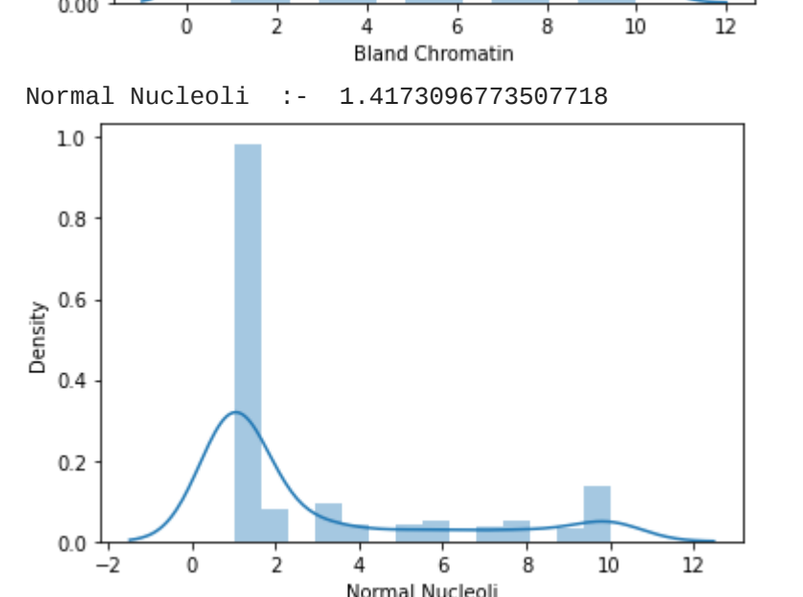

```
Bare Nuclei  :-  0.9878400612076661
```



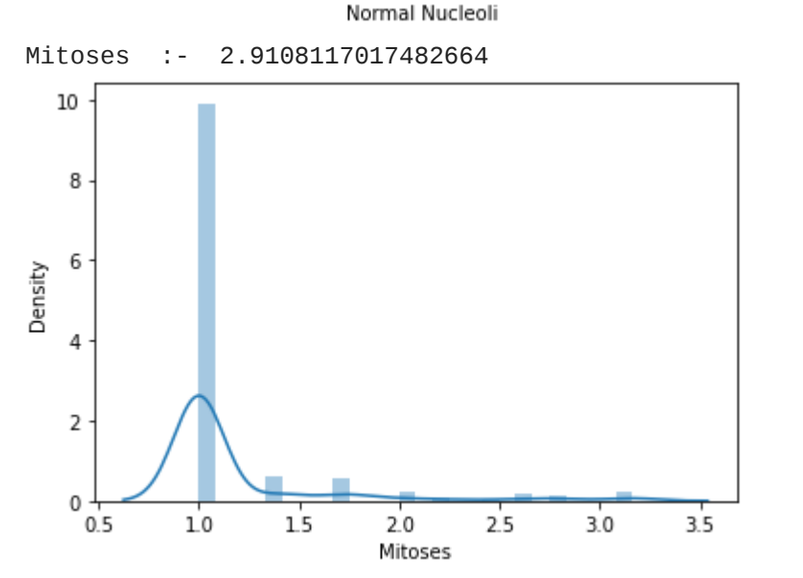

```
Bland Chromatin  :-  1.0928635746903066
```



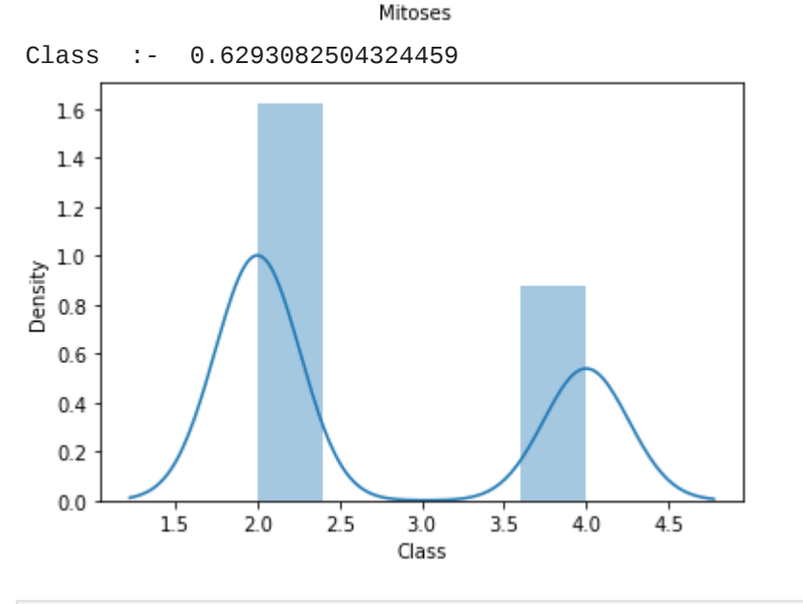

```
Normal Nucleoli  :-  1.4173096773507718
```




```
Mitoses  :-  2.9108117017482664
```




```
Class  :-  0.6293082504324459
```



In [61]:

```python
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [63]:

```python
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
from sklearn.model_selection import train_test_split ,cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier , AdaBoostClassifier   ,GradientBoostingClassifier
from xgboost import XGBClassifier
```

In [77]:

```python
xtrain,xtest,ytrain,ytest=train_test_split(x,y,random_state=0,test_size=0.30)
```

In [78]:

```python
models=[('logistic Regression         :- ',LogisticRegression()),
        ('KNeighbors Classifier       :- ' ,KNeighborsClassifier()),
        ('Deceision Tree              :- ',DecisionTreeClassifier()),
        ('Support vectore Classifier  :- ',SVC()),
        ('Random Forest Clasifier     :- ',RandomForestClassifier()),
        ('Ada Boosting Classifier     :- ',AdaBoostClassifier()),
        ('Gradient Boosting Classifier :- ' ,GradientBoostingClassifier()),
        ('Xtreame Boosting Classifier  :- ',XGBClassifier())

       ]
```

In [79]:

```python
accuracy=[]
for name,model in models:
    print(name)
    model.fit(xtrain,ytrain)
    ypred=model.predict(xtest)
    print('Confusion matrix  :-  ')
    print(confusion_matrix(ytest,ypred))
    print(classification_report(ytest,ypred))
    accuracy.append(accuracy_score(ytest,ypred))
```

```
logistic Regression         :-
Confusion matrix  :-
[[126   4]
 [  7  68]]
              precision    recall  f1-score   support

           2       0.95      0.97      0.96       130
           4       0.94      0.91      0.93        75

    accuracy                           0.95       205
   macro avg       0.95      0.94      0.94       205
weighted avg       0.95      0.95      0.95       205

KNeighbors Classifier       :-
Confusion matrix  :-
[[127   3]
 [  5  70]]
              precision    recall  f1-score   support

           2       0.96      0.98      0.97       130
           4       0.96      0.93      0.95        75

    accuracy                           0.96       205
   macro avg       0.96      0.96      0.96       205
weighted avg       0.96      0.96      0.96       205

Deceision Tree              :-
Confusion matrix  :-
[[125   5]
 [  8  67]]
              precision    recall  f1-score   support

           2       0.94      0.96      0.95       130
           4       0.93      0.89      0.91        75

    accuracy                           0.94       205
   macro avg       0.94      0.93      0.93       205
weighted avg       0.94      0.94      0.94       205

Support vectore Classifier  :-
Confusion matrix  :-
[[124   6]
 [  5  70]]
              precision    recall  f1-score   support

           2       0.96      0.95      0.96       130
           4       0.92      0.93      0.93        75

    accuracy                           0.95       205
   macro avg       0.94      0.94      0.94       205
weighted avg       0.95      0.95      0.95       205

Random Forest Clasifier     :-
Confusion matrix  :-
[[126   4]
 [  4  71]]
              precision    recall  f1-score   support

           2       0.97      0.97      0.97       130
           4       0.95      0.95      0.95        75

    accuracy                           0.96       205
   macro avg       0.96      0.96      0.96       205
weighted avg       0.96      0.96      0.96       205

Ada Boosting Classifier     :-
Confusion matrix  :-
[[126   4]
 [  8  67]]
              precision    recall  f1-score   support

           2       0.94      0.97      0.95       130
           4       0.94      0.89      0.92        75

    accuracy                           0.94       205
   macro avg       0.94      0.93      0.94       205
weighted avg       0.94      0.94      0.94       205

Gradient Boosting Classifier :-
Confusion matrix  :-
[[126   4]
 [  8  67]]
              precision    recall  f1-score   support

           2       0.94      0.97      0.95       130
           4       0.94      0.89      0.92        75

    accuracy                           0.94       205
   macro avg       0.94      0.93      0.94       205
weighted avg       0.94      0.94      0.94       205

Xtreame Boosting Classifier  :-
[22:08:36] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'er
ror' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
Confusion matrix  :-
[[125   5]
 [  6  69]]
              precision    recall  f1-score   support

           2       0.95      0.96      0.96       130
           4       0.93      0.92      0.93        75

    accuracy                           0.95       205
   macro avg       0.94      0.94      0.94       205
weighted avg       0.95      0.95      0.95       205
```

In [80]:

```python
print('Mean accuracy of all the models   :-  ',np.mean(accuracy)*100)
```

```
Mean accuracy of all the models   :-   94.7560975609756
```

from the above information we Can say that KNeighbors Classifier is the best model among all of them

In [82]:

```python
knn=KNeighborsClassifier()
knn.fit(xtrain,ytrain)
ypred=knn.predict(xtest)
print('Confusion Matrix')
print(confusion_matrix(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
Confusion Matrix
[[127   3]
 [  5  70]]
              precision    recall  f1-score   support

           2       0.96      0.98      0.97       130
           4       0.96      0.93      0.95        75

    accuracy                           0.96       205
   macro avg       0.96      0.96      0.96       205
weighted avg       0.96      0.96      0.96       205
```

In [85]:

```python
print('trainig score  :-  ',knn.score(xtrain,ytrain)*100)
print('testing score   :-  ',knn.score(xtest,ytest)*100)
```

```
trainig score  :-   97.90794979079497
testing score   :-   96.09756097560975
```

In [89]:

```python
c=cross_val_score(knn,x,y,cv=10)
```

In [94]:

```python
print('Cross Validation Value :-  ',c.mean()*100)
```

```
Cross Validation Value :-   97.22293265132139
```

In [ ]: