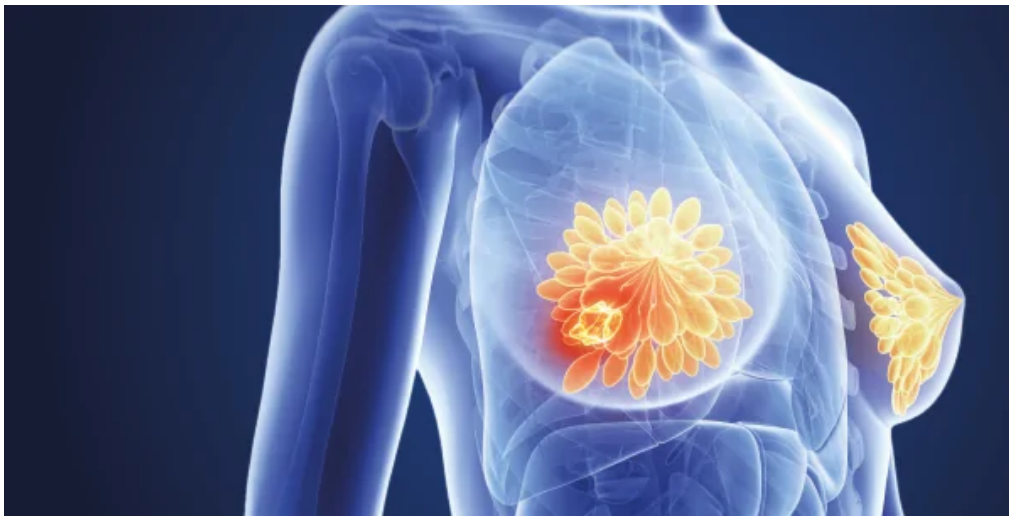


The aim of this is to successfully be able to classify cancer as being Benign or Mali
attributes

To know more about the dataset used in the research visit here (https://github.com/
DataDetective/Top_11_African_Countries/blob/master/aged_15plus_employment_

In [1]:
```
#importing the libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
import seaborn as sns

%matplotlib inline
```

In [2]:
```
data = pd.read_csv('breastCancer.csv')
```

In [3]:
```
data.head(5)
```

	id	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial
0	1000025	5	1	1	1	2
1	1002945	5	4	4	5	7
2	1015425	3	1	1	1	2
3	1016277	6	8	8	1	3
4	1017023	4	1	1	3	2

Data Preprocessing

In [4]:
```
data['class'].value_counts()
```

```
2    458
4    241
Name: class, dtype: int64
```

In [5]:
```
data.dtypes
```

```
id                   int64
clump_thickness      int64
size_uniformity      int64
shape_uniformity     int64
marginal_adhesion    int64
epithelial_size      int64
bare_nucleoli       object
bland_chromatin      int64
normal_nucleoli      int64
mitoses              int64
class                int64
dtype: object
```

In [6]:
```
data[data['bare_nucleoli']=='?'] #we have a lot of question marks thats making the variable
```

	id	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithel
23	1057013	8	4	5	1	2
40	1096800	6	6	6	9	6
139	1183246	1	1	1	1	1
145	1184840	1	1	3	1	2
158	1193683	1	1	2	1	3
164	1197510	5	1	1	1	2
235	1241232	3	1	4	1	2
249	169356	3	1	1	1	2
275	432809	3	1	3	1	2
292	563649	8	8	8	1	2
294	606140	1	1	1	1	2
297	61634	5	4	3	1	2
315	704168	4	6	5	6	7
321	733639	3	1	1	1	2
411	1238464	1	1	1	1	1
617	1057067	1	1	1	1	1

In [7]:
```
data[data['bare_nucleoli']=='?'].sum()
```

```
id                          13721250
clump_thickness                   54
size_uniformity                   39
shape_uniformity                  46
marginal_adhesion                 29
epithelial_size                   39
bare_nucleoli        ????????????????
bland_chromatin                   50
normal_nucleoli                   44
mitoses                           16
class                             36
dtype: object
```

In [8]:
```
df = data.replace('?',np.nan)
```

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

	id	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial
0	1000025	5	1	1	1	2
1	1002945	5	4	4	5	7
2	1015425	3	1	1	1	2
3	1016277	6	8	8	1	3
4	1017023	4	1	1	3	2

In [10]:
```
df['bare_nucleoli'].unique()
```

```
array(['1', '10', '2', '4', '3', '9', '7', nan, '5', '8', '6'],
      dtype=object)
```

In [11]:
```
#replacing the nan values with median
df.median()
```

```
id                   1171710.0
clump_thickness            4.0
size_uniformity            1.0
shape_uniformity           1.0
marginal_adhesion          1.0
epithelial_size            2.0
bare_nucleoli              1.0
bland_chromatin            3.0
normal_nucleoli            1.0
mitoses                    1.0
class                      2.0
dtype: float64
```

In [12]:
```
df = df.fillna(df.median())
```

In [13]:
```
df.bare_nucleoli.unique()
```

```
array(['1', '10', '2', '4', '3', '9', '7', 1.0, '5', '8', '6'],
      dtype=object)
```

In [14]:
```
df.bare_nucleoli = df.bare_nucleoli.astype(int)
```

In [15]:
```
df.dtypes
```

```
id                   int64
clump_thickness      int64
size_uniformity      int64
shape_uniformity     int64
marginal_adhesion    int64
epithelial_size      int64
bare_nucleoli        int32
bland_chromatin      int64
normal_nucleoli      int64
mitoses              int64
class                int64
dtype: object
```

Exploratory Data Analysis

In [16]:
```
# dropping column id
df.drop('id',axis=1,inplace=True)
```

In [17]:
```
df.describe().T
```

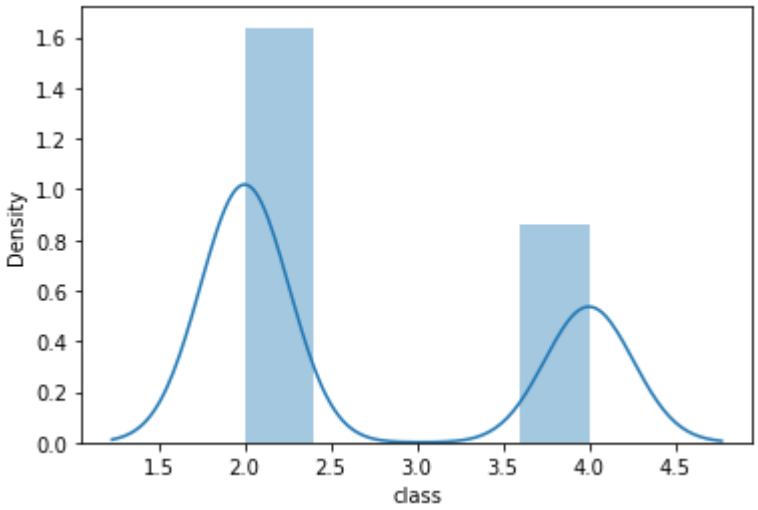
	count	mean	std	min	25%	50%	75%	max
clump_thickness	699.0	4.417740	2.815741	1.0	2.0	4.0	6.0	10.0
size_uniformity	699.0	3.134478	3.051459	1.0	1.0	1.0	5.0	10.0
shape_uniformity	699.0	3.207439	2.971913	1.0	1.0	1.0	5.0	10.0
marginal_adhesion	699.0	2.806867	2.855379	1.0	1.0	1.0	4.0	10.0
epithelial_size	699.0	3.216023	2.214300	1.0	2.0	2.0	4.0	10.0
bare_nucleoli	699.0	3.486409	3.621929	1.0	1.0	1.0	5.0	10.0
bland_chromatin	699.0	3.437768	2.438364	1.0	2.0	3.0	5.0	10.0
normal_nucleoli	699.0	2.866953	3.053634	1.0	1.0	1.0	4.0	10.0
mitoses	699.0	1.589413	1.715078	1.0	1.0	1.0	1.0	10.0
class	699.0	2.689557	0.951273	2.0	2.0	2.0	4.0	4.0

Bivariate Analysis

In [18]:
```
sns.distplot(df['class'])
```

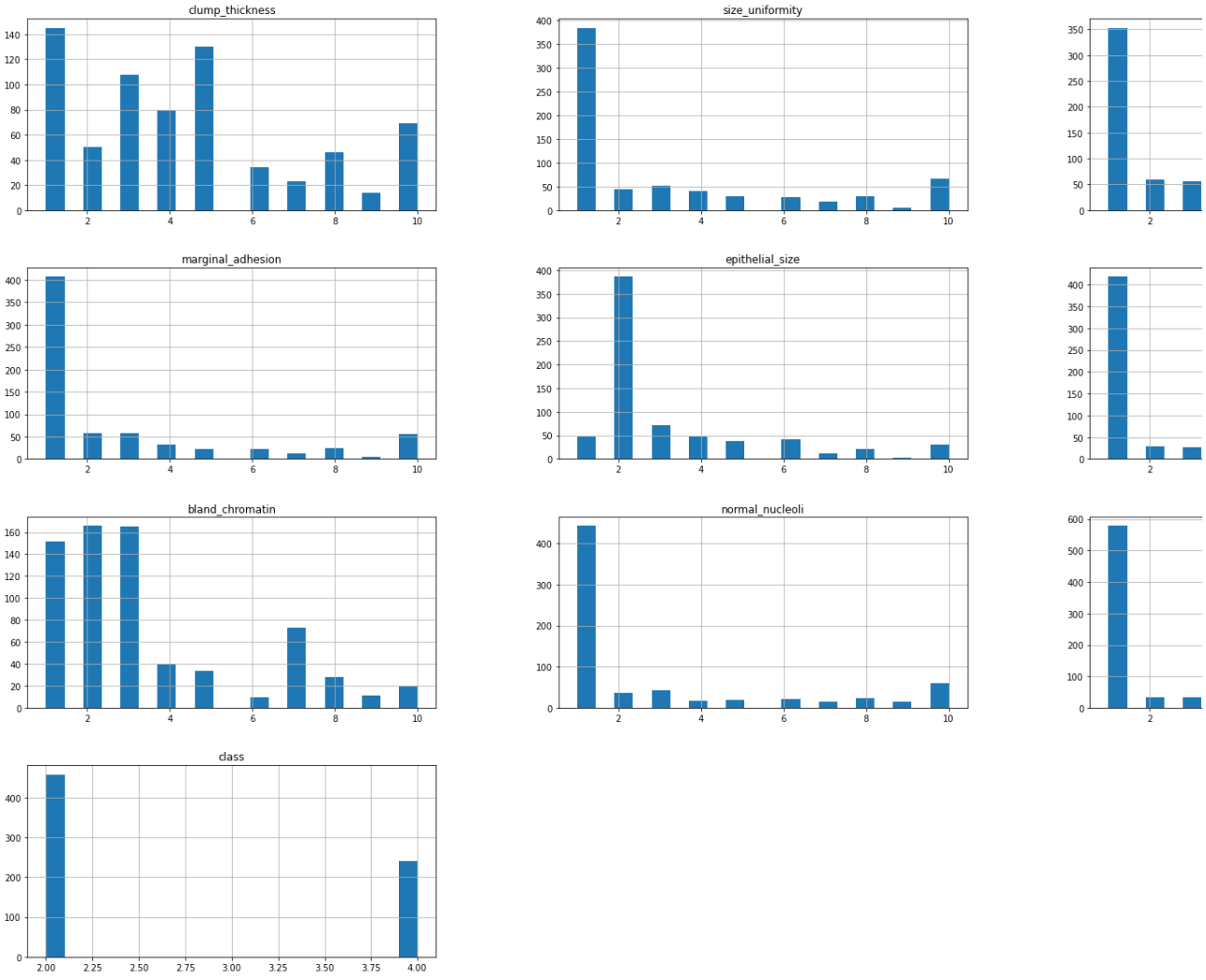
```
C:\Users\Folayan Tobi\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning
ction and will be removed in a future version. Please adapt your code to use either `displot`
milar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='class', ylabel='Density'>
```



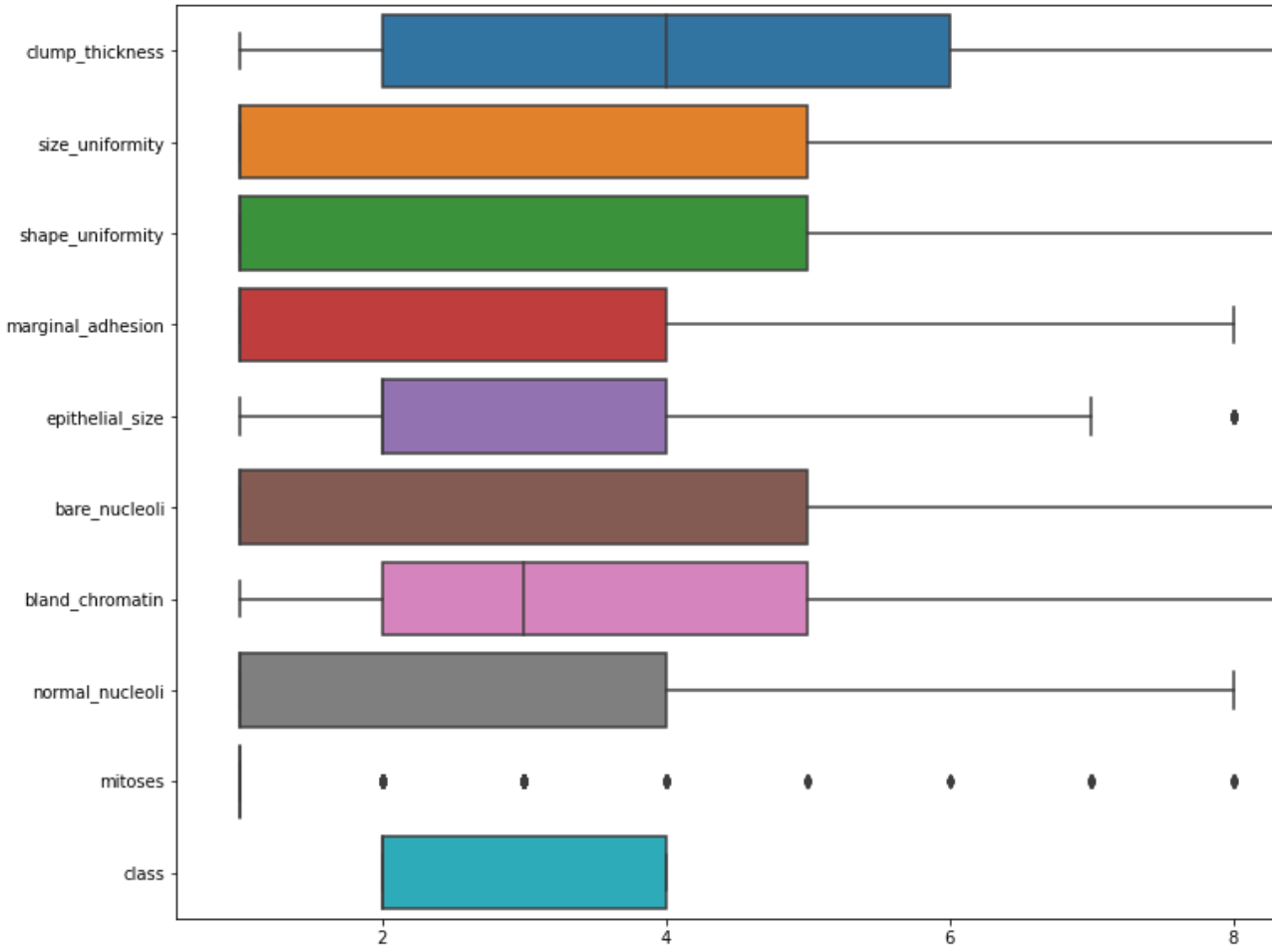
Multivariate Analysis

In [19]:
```
df.hist(bins=20, figsize=(30,30), layout=(6,3));
```



In [21]:
```
plt.figure(figsize = (15,10))
sns.boxplot(data=df, orient='h')
```

```
<AxesSubplot:>
```

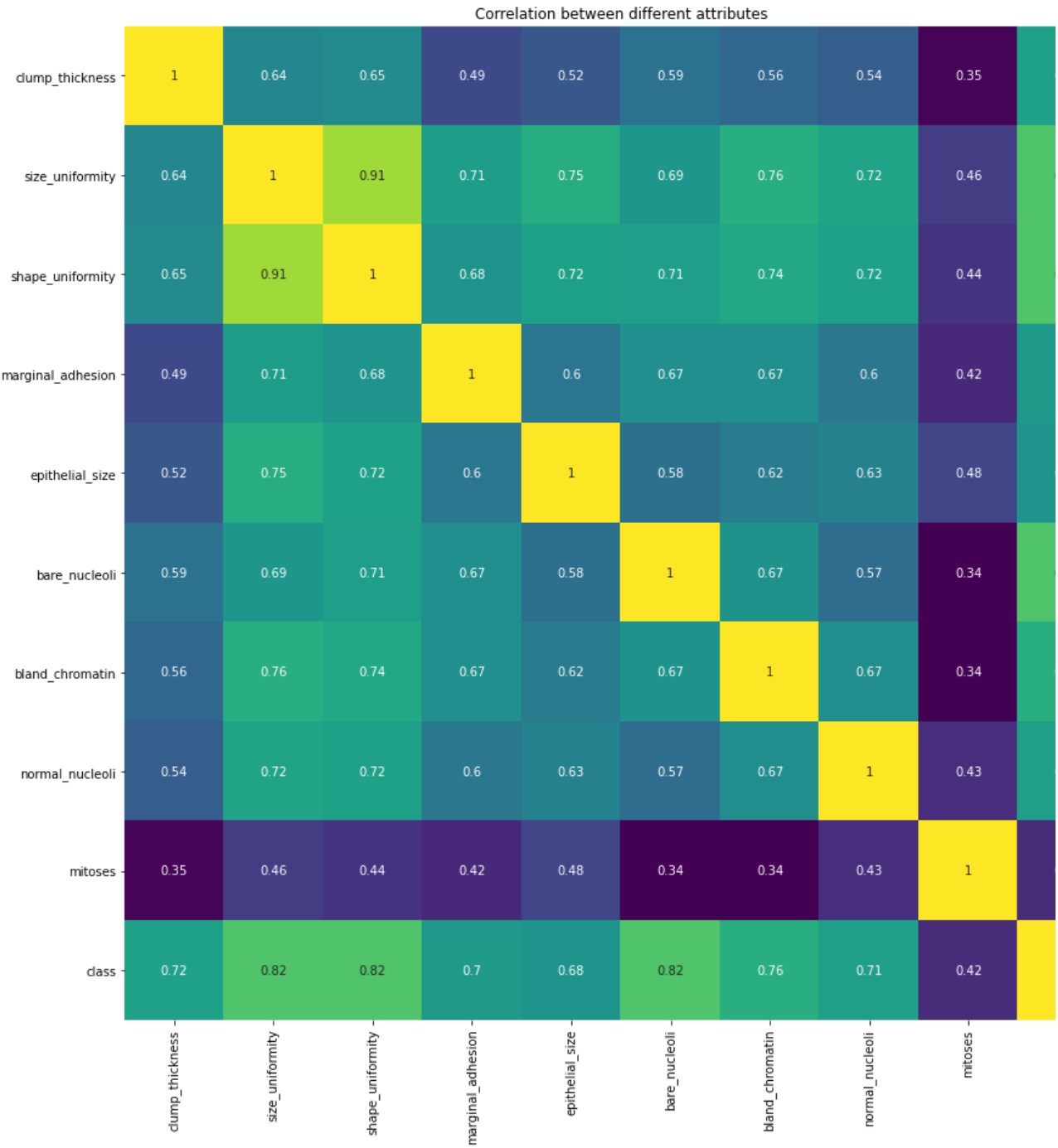


In [22]:
```
df.corr()
```

	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	ep
clump_thickness	1.000000	0.644913	0.654589	0.486356	0.5
size_uniformity	0.644913	1.000000	0.906882	0.705582	0.7
shape_uniformity	0.654589	0.906882	1.000000	0.683079	0.7
marginal_adhesion	0.486356	0.705582	0.683079	1.000000	0.5
epithelial_size	0.521816	0.751799	0.719668	0.599599	1.0
bare_nucleoli	0.590008	0.686673	0.707474	0.666971	0.5
bland_chromatin	0.558428	0.755721	0.735948	0.666715	0.6
normal_nucleoli	0.535835	0.722865	0.719446	0.603352	0.6
mitoses	0.350034	0.458693	0.438911	0.417633	0.4
class	0.716001	0.817904	0.818934	0.696800	0.6

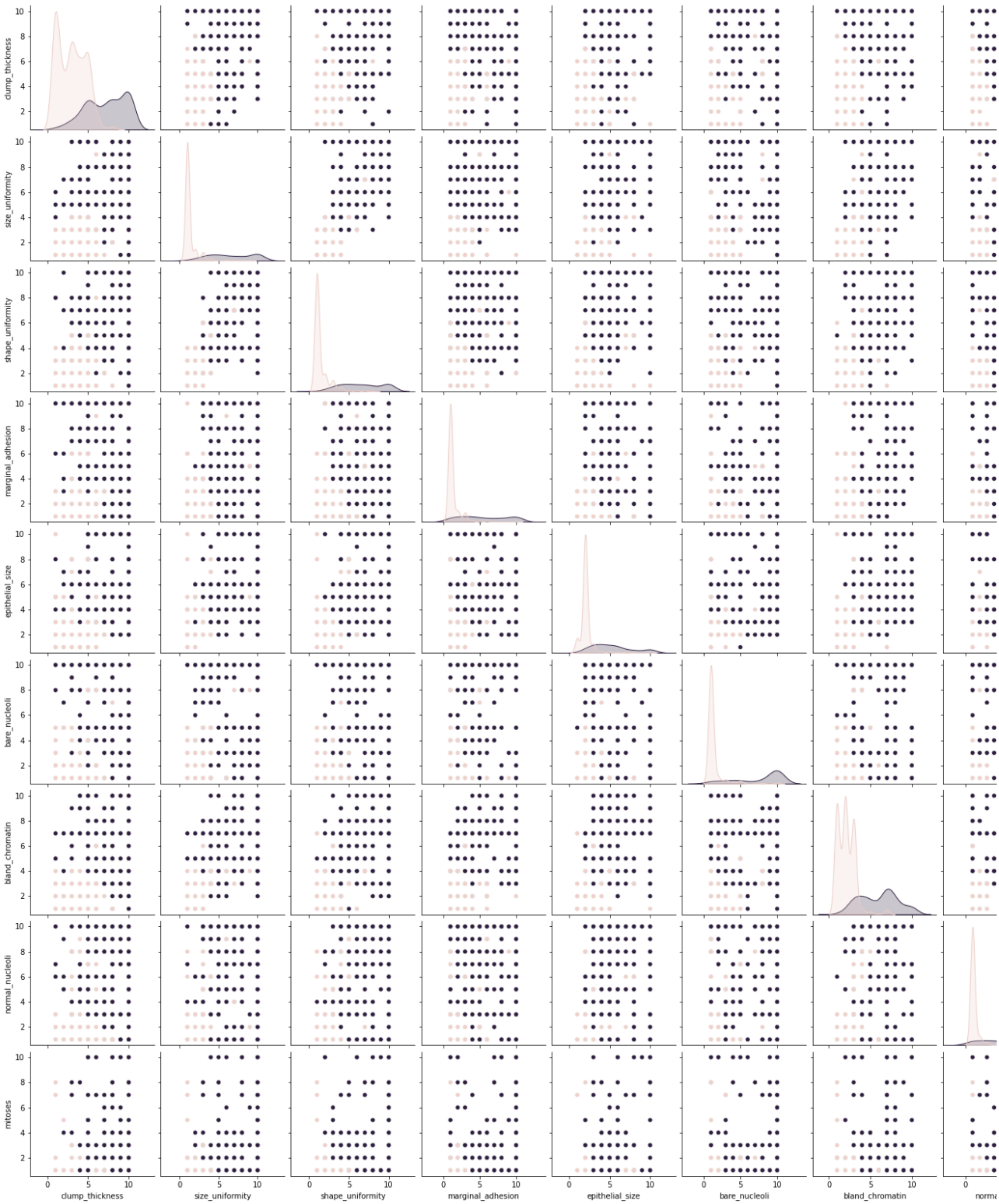
In [23]:
```
plt.figure(figsize = (35,15))
sns.heatmap(df.corr(), vmax=1, square = True, annot =True, cmap ='viridis')
plt.title('Correlation between different attributes')
```

```
Text(0.5, 1.0, 'Correlation between different attributes')
```



In [56]:
```
sns.pairplot(df, diag_kind='kde', hue='class')
```

```
<seaborn.axisgrid.PairGrid at 0x27f303e7c70>
```



Building the Model

In [25]:
```
x = df.drop('class', axis=True)
y = df['class']
```

In [26]:
```
from sklearn.model_selection import train_test_split
```

In [28]:
```
# splitting Ratio 70:30

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.30,
                                                    random_state=1)
```

KNN

In [29]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [30]:
```
KNN = KNeighborsClassifier(n_neighbors=5, weights='distance')
```

In [31]:
```
KNN.fit(X_train, y_train)
```

```
KNeighborsClassifier(weights='distance')
```

In [32]:
```
KNN_prediction = KNN.predict(X_test)
KNN_prediction
```

```
C:\Users\Folayan Tobi\anaconda3\lib\site-packages\sklearn\base.py:441: UserWarning: X does not
NeighborsClassifier was fitted with feature names
  warnings.warn(
```

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

In [43]:
```
from scipy.stats import zscore

print('KNeighborsClassifier Algorithm predicts at {0:.4g}%'.format(KNN.score(X_test,y_test
```

```
KNeighborsClassifier Algorithm predicts at 97.62%
```

```
C:\Users\Folayan Tobi\anaconda3\lib\site-packages\sklearn\base.py:441: UserWarning: X does not
NeighborsClassifier was fitted with feature names
  warnings.warn(
```

SVC

In [40]:
```
from sklearn.svm import SVC

svc = SVC(gamma= 0.025, C=3)
svc.fit(X_train, y_train)
```

```
SVC(C=3, gamma=0.025)
```

In [41]:
```
svc_prediction = svc.predict(X_test)
svc_prediction
```

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

In [44]:
```
print('SVC Algorithm predicts at {0:.4g}%'.format(svc.score(X_test,y_test)*100));
```

```
SVC Algorithm predicts at 98.1%
```

In [45]:
```
KNN_prediction= pd.DataFrame(KNN_prediction)
svc_prediction= pd.DataFrame(svc_prediction)

predictions = pd.concat([KNN_prediction,svc_prediction],axis=1)
```

In [46]:
```
predictions.columns = ['KNN_prediction','svc_prediction']
```

In [47]:
```
predictions
```

	KNN_prediction	svc_prediction
0	2	2
1	2	2
2	2	2
3	4	4
4	2	2
...	...	...
205	4	4
206	4	4
207	2	2
208	2	2
209	2	2

210 rows × 2 columns

Classification Report

In [48]:
```
from sklearn.metrics import classification_report
```

In [50]:
```
print("KNN classification report")
print('_________'*5)
classification_report(y_test,KNN_prediction)
```

```
 KNN classification report
 ---------------------------------------------


 '              precision    recall  f1-score   support\n\n           2       0.97      0.99
 0.99      0.95      0.97        73\n\n    accuracy                           0.98       210\n
 0.97       210\nweighted avg       0.98      0.98      0.98       210\n'
```

In [51]:
```
print("Support Vector Classification report")
print('_________'*5)
classification_report(y_test,svc_prediction)
```

```
 Support Vector Classification report
 ---------------------------------------------


 '              precision    recall  f1-score   support\n\n           2       0.99      0.99
 0.97      0.97      0.97        73\n\n    accuracy                           0.98       210\n
 0.98       210\nweighted avg       0.98      0.98      0.98       210\n'
```

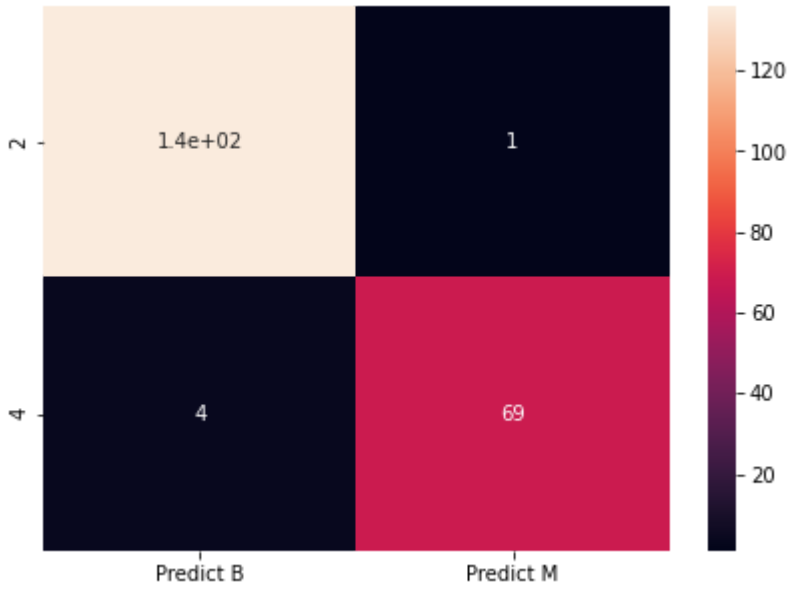
In [53]:
```
from sklearn import metrics

print('Confusion matrix for KNN')
cm = metrics.confusion_matrix(y_test, KNN_prediction, labels=[2,4])

df_cm = pd.DataFrame(cm, index=[i for i in [2,4]], columns=[i for i in [
    'Predict B','Predict M'
]])

plt.figure(figsize=(7,5))
sns.heatmap(df_cm, annot=True);
```

```
Confusion matrix for KNN
```

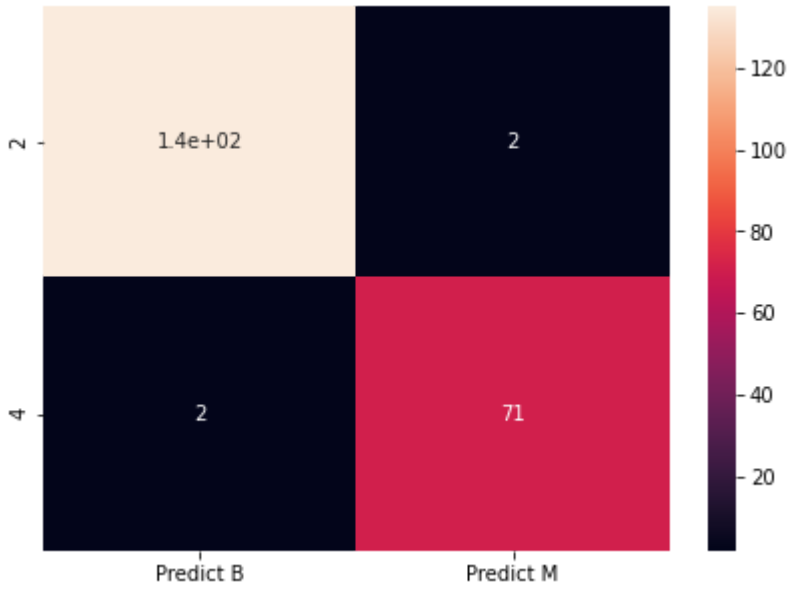


In [54]:
```
print('Confusion matrix for SVC')
cm = metrics.confusion_matrix(y_test, svc_prediction, labels=[2,4])

df_cm = pd.DataFrame(cm, index=[i for i in [2,4]], columns=[i for i in [
    'Predict B','Predict M'
]])

plt.figure(figsize=(7,5))
sns.heatmap(df_cm, annot=True);
```

```
Confusion matrix for SVC
```



In [ ]: