Source:

Creator:

Dr. WIlliam H. Wolberg (physician)

University of Wisconsin Hospitals

Madison, Wisconsin, USA

Location:

UCI data repository

Kaggle data repository

Attribute Information:

1. Sample code number: id number
2. Clump Thickness: 1 - 10
3. Uniformity of Cell Size: 1 - 10
4. Uniformity of Cell Shape: 1 - 10
5. Marginal Adhesion: 1 - 10
6. Single Epithelial Cell Size: 1 - 10
7. Bare Nuclei: 1 - 10
8. Bland Chromatin: 1 - 10
9. Normal Nucleoli: 1 - 10
10. Mitoses: 1 - 10
11. Class: (2 for benign, 4 for malignant)

Malignant==> Cancerous

Benign==> Not Cancerous (Healthy)

Background

All of our bodies are composed of cells. The human body has about 100 trillion cells within it. And usually those cells behave in a certain way. However, occasionally, one of these 100 trillion cells, behave in a different way and keeps dividing and pushes the other cells around it out of the way. That cell stops observing the rules of the tissue within which it is located and begins to move out of its normal position and starts invading into the tissues around it and sometimes entering the bloodstream and becoming is called a metastasis.

In summary, as we grow older,throughout a lifetime, we go through this knid of situation where a particular kind of gene is mutated where the protein that it makes is abnormal and drives the cell to behave in a different way that we call cancer.

This is what Dr. WIlliam H. Wolberg was observing and put together this dataset.

Can we predict whether a cell is Malignant or Benign?

In [162]:
```
# !pip install pyforest
# from pyforest import*
# lazy_imports()
```

In [163]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [164]:
```
data=pd.read_csv("breastcancer.csv")
```

In [165]:
```
data.head()
```

Out[165]:

	id	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bare_n
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 pre-processing

In [166]:
```
data['class'].value_counts()
```

Out[166]:
```
2    458
4    241
Name: class, dtype: int64
```

In [167]:
```
data.dtypes #checking the data types of each column
```

Out[167]:
```
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 [168]:
```
data['bare_nucleoli'] #let's inspect the 'bare_nucleoli' column
```

Out[168]:
```
0       1
1      10
2       2
3       4
4       1
5      10
6      10
7       1
8       1
9       1
10      1
11      1
12      3
13      3
14      9
15      1
16      1
17      1
18     10
19      1
20     10
21      7
22      1
23      ?
24      1
25      7
26      1
27      1
28      1
29      1
       ..
669     5
670     8
671     1
672     1
673     1
674     1
675     1
676     1
677     1
678     1
679     1
680    10
681    10
682     1
683     1
684     1
685     1
686     1
687     1
688     1
689     1
690     1
691     5
692     1
693     1
694     2
695     1
696     3
697     4
698     5
Name: bare_nucleoli, Length: 699, dtype: object
```

In [169]:
```
data[data['bare_nucleoli']=='?'] #checking the presence of '?' in the 'bare_nucleoli' column
```

Out[169]:

	id	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bare
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 [170]:
```
data[data['bare_nucleoli']=='?'].sum() # alternatively
```

Out[170]:
```
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
```

Alternatively

Using the isdigit() function

In [171]:
```
digits_in_bare_nucleoli= pd.DataFrame(data.bare_nucleoli.str.isdigit())
digits_in_bare_nucleoli
```

Out[171]:

	bare_nucleoli
0	True
1	True
2	True
3	True
4	True
5	True
6	True
7	True
8	True
9	True
10	True
11	True
12	True
13	True
14	True
15	True
16	True
17	True
18	True
19	True
20	True
21	True
22	True
23	False
24	True
25	True
26	True
27	True
28	True
29	True
...	...
669	True
670	True
671	True
672	True
673	True
674	True
675	True
676	True
677	True
678	True

	bare_nucleoli
679	True
680	True
681	True
682	True
683	True
684	True
685	True
686	True
687	True
688	True
689	True
690	True
691	True
692	True
693	True
694	True
695	True
696	True
697	True
698	True

699 rows × 1 columns

In [172]:
```
#df[digits_in_hp['horsepower'] == False]
data[digits_in_bare_nucleoli['bare_nucleoli']== False]
```

Out[172]:

	id	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bare
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	

Let us replace these missing values with NaN

In [173]:
```
df= data.replace('?', np.nan)
```

In [ ]:
```

```

In [174]:
```
df.bare_nucleoli
```

Out[174]:
```
0        1
1       10
2        2
3        4
4        1
5       10
6       10
7        1
8        1
9        1
10       1
11       1
12       3
13       3
14       9
15       1
16       1
17       1
18      10
19       1
20      10
21       7
22       1
23     NaN
24       1
25       7
26       1
27       1
28       1
29       1
      ...
669      5
670      8
671      1
672      1
673      1
674      1
675      1
676      1
677      1
678      1
679      1
680     10
681     10
682      1
683      1
684      1
685      1
686      1
687      1
688      1
689      1
690      1
691      5
692      1
693      1
694      2
695      1
696      3
697      4
698      5
Name: bare_nucleoli, Length: 699, dtype: object
```

In [ ]:

In [175]:
```
df.median()
```

Out[175]:
```
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 [176]:
```
df.head()
```

Out[176]:

	id	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bare_n
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 [177]:
```
df = df.fillna(df.median())
```

In [178]:
```
df.dtypes
```

Out[178]:
```
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 [179]:
```
df.bare_nucleoli
```

Out[179]:
```
0       1
1      10
2       2
3       4
4       1
5      10
6      10
7       1
8       1
9       1
10      1
11      1
12      3
13      3
14      9
15      1
16      1
17      1
18     10
19      1
20     10
21      7
22      1
23      1
24      1
25      7
26      1
27      1
28      1
29      1
       ..
669     5
670     8
671     1
672     1
673     1
674     1
675     1
676     1
677     1
678     1
679     1
680    10
681    10
682     1
683     1
684     1
685     1
686     1
687     1
688     1
689     1
690     1
691     5
692     1
693     1
694     2
695     1
696     3
697     4
698     5
Name: bare_nucleoli, Length: 699, dtype: object
```

In [180]:
```
df.dtypes
```

Out[180]:
```
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 [181]:
```
df['bare_nucleoli'] = df['bare_nucleoli'].astype('int64')
```

In [182]:
```
df.dtypes
```

Out[182]:
```
id                   int64
clump_thickness      int64
size_uniformity      int64
shape_uniformity     int64
marginal_adhesion    int64
epithelial_size      int64
bare_nucleoli        int64
bland_chromatin      int64
normal_nucleoli      int64
mitoses              int64
class                int64
dtype: object
```

Exploratory Data Analysis

In [183]:
```
#dropping the index of the dataset

df.drop('id', axis=1, inplace=True)
```

In [184]:
```
df.head()
```

Out[184]:

	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bare_nucleoli	b
0	5	1	1	1	2	1	
1	5	4	4	5	7	10	
2	3	1	1	1	2	2	
3	6	8	8	1	3	4	
4	4	1	1	3	2	1	

In [185]:
```
data.head()
```

Out[185]:

	id	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bare_r
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 [186]:
```
df.describe().T
```

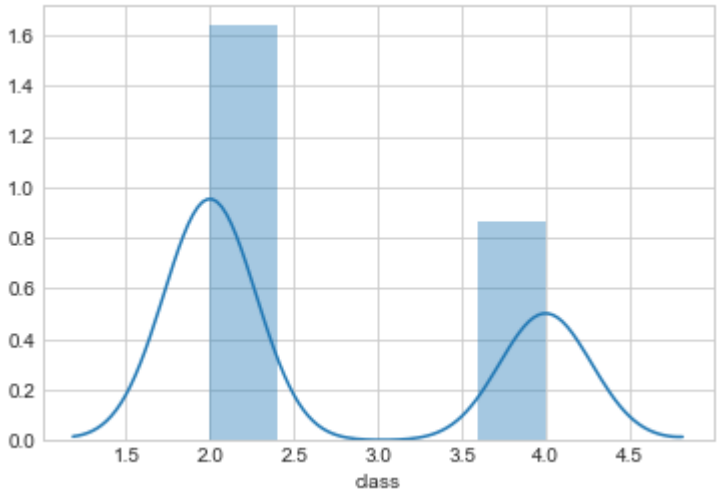
Out[186]:

	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 Data Analysis

In [187]:
```
sns.distplot(df['class'])
```

Out[187]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a22421438>
```



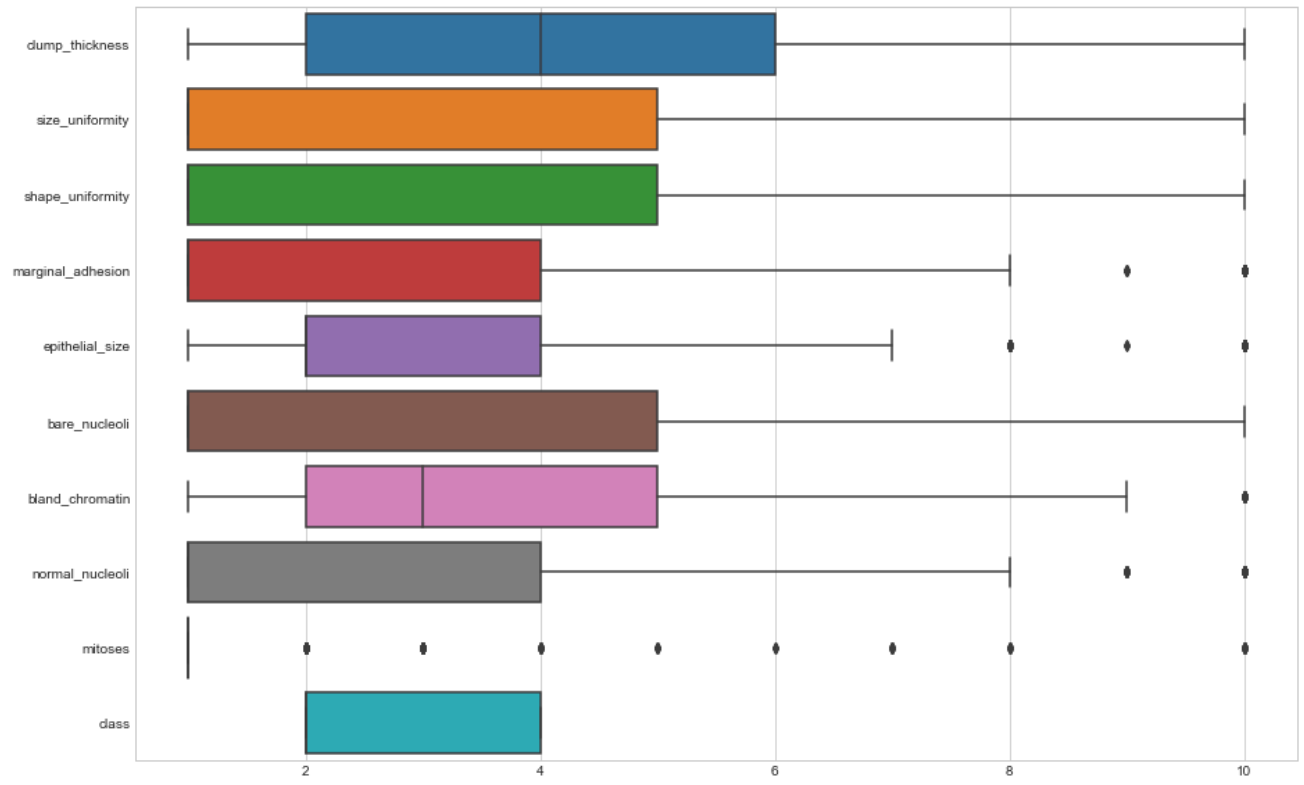
Each is even distributed and there are beningn than malignant

Multivariate Data Analysis

In [188]:
```
df.hist(bins=20, figsize=(30,30), layout=(6,3));
```

In [189]:
```
plt.figure(figsize= (15,10))
sns.boxplot(data=df,orient="h")
```

Out[189]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a22994780>
```



Let's See the Correlation among these attributes

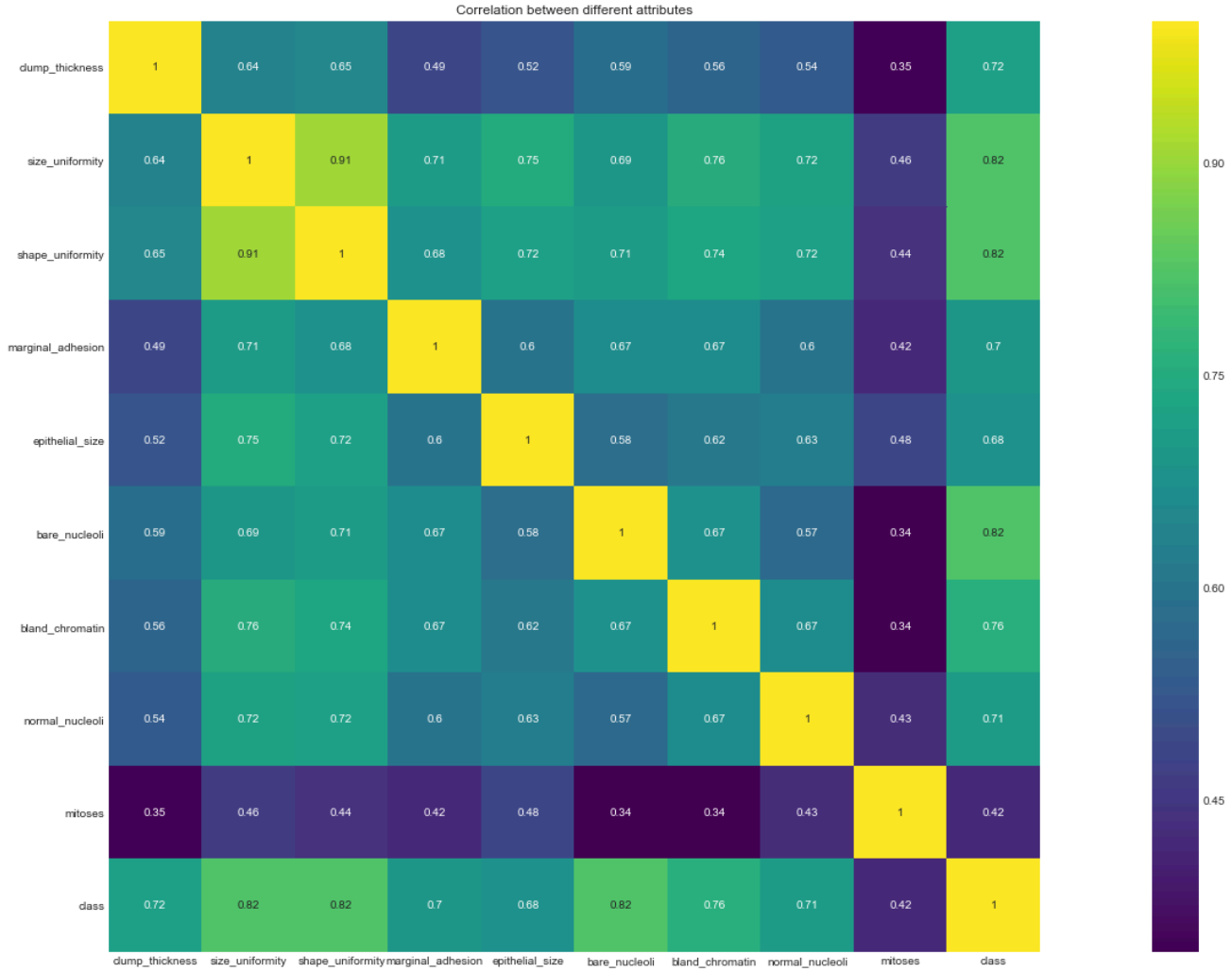
In [190]:
```
df.corr()
```

Out[190]:

	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size
clump_thickness	1.000000	0.644913	0.654589	0.486356	0.521816
size_uniformity	0.644913	1.000000	0.906882	0.705582	0.751799
shape_uniformity	0.654589	0.906882	1.000000	0.683079	0.719668
marginal_adhesion	0.486356	0.705582	0.683079	1.000000	0.599599
epithelial_size	0.521816	0.751799	0.719668	0.599599	1.000000
bare_nucleoli	0.590008	0.686673	0.707474	0.666971	0.583701
bland_chromatin	0.558428	0.755721	0.735948	0.666715	0.616102
normal_nucleoli	0.535835	0.722865	0.719446	0.603352	0.628881
mitoses	0.350034	0.458693	0.438911	0.417633	0.479101
class	0.716001	0.817904	0.818934	0.696800	0.682785

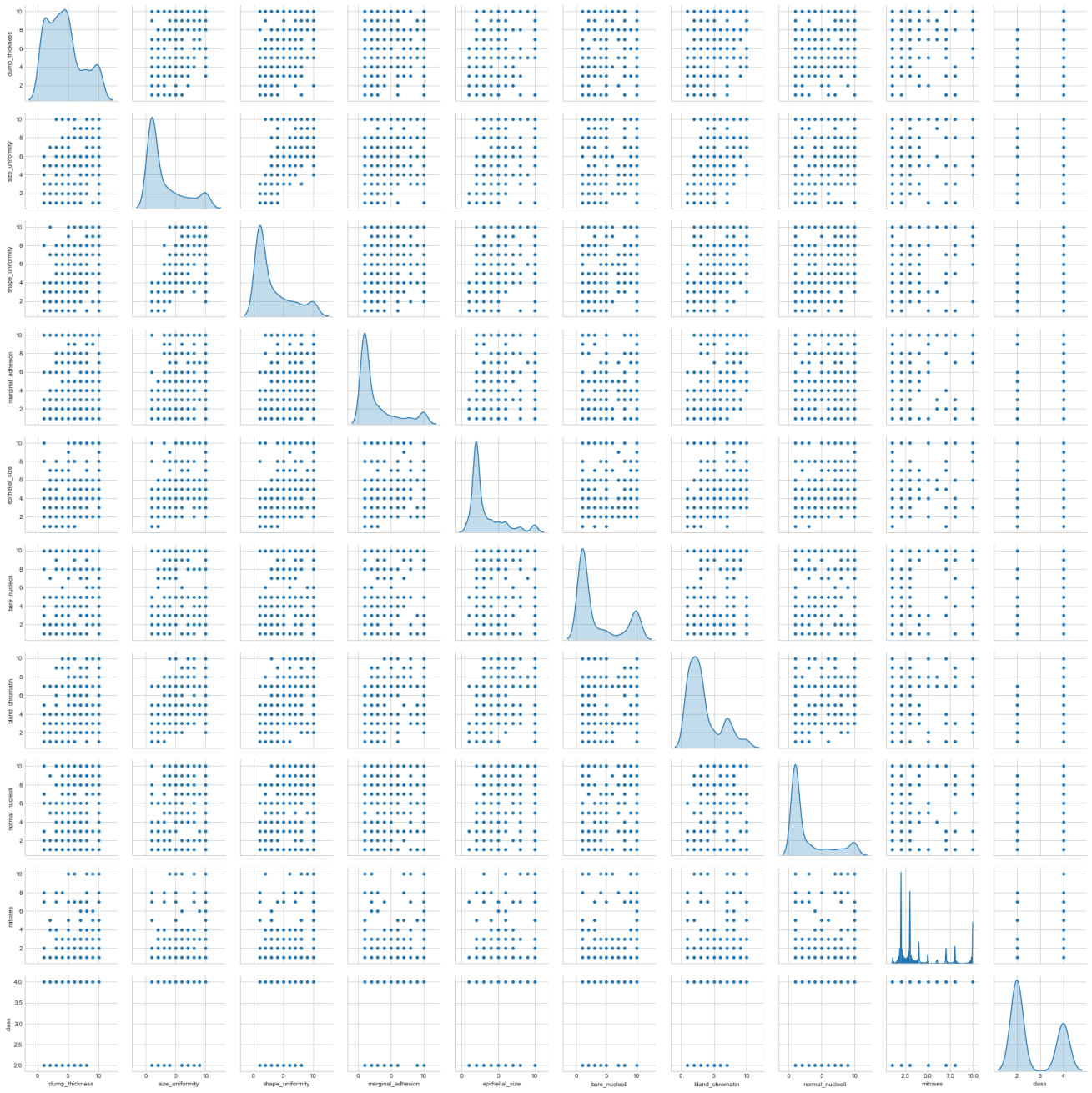
In [191]:
```
#Heatmap of the correlation between the indepent attributes

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



In [192]:
```
#Pairplot of the correlation/distribution between various independent attributes
sns.pairplot(df, diag_kind="kde")
```

Out[192]: <seaborn.axisgrid.PairGrid at 0x1a229bd668>



In [ ]:

Building Our Model

In [193]:
```
# Dividing our dataset into training and testing set

X = df.drop('class', axis=1)  #selecting all the attributes except the class attribute
y = df['class'] #selecting class attribute.
```

In [ ]:

In [194]:
```
#Splitting our data into 70:30
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, rando
m_state=1)
```

In [ ]:
```

```

KNeighborsClassifier

In [195]:
```
from sklearn.neighbors import KNeighborsClassifier

KNN = KNeighborsClassifier(n_neighbors= 5 , weights = 'distance' )
```

In [196]:
```
# Call Nearest Neighbour algorithm

KNN.fit(X_train, y_train)
```

Out[196]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='distance')
```

In [197]:
```
predicted_1 = KNN.predict(X_test)
predicted_1
```

Out[197]:
```
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, 2, 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])
```

In [198]:
```
from scipy.stats import zscore

print('KNeighborsClassifier Agorithm is predicting at {0:.2g}%'.format(KNN.scor
e(X_test, y_test)*100))
```

```
KNeighborsClassifier Agorithm is predicting at 97%
```

In [ ]:
```

```

Support Vector Machine

In [199]:
```
from sklearn.svm import SVC


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

Out[199]:
```
SVC(C=3, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.025, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [200]:
```
predicted_2 = svc.predict(X_test)
predicted_2
```

Out[200]:
```
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])
```

In [201]:
```
print('SupportVectorClassifier Agorithm is predicting at {0:.2g}%'.format(svc.score(X_test, y_test)*100))
```

```
SupportVectorClassifier Agorithm is predicting at 98%
```

In [202]:
```
knnPredictions=pd.DataFrame(predicted_1)
svcPredictions=pd.DataFrame(predicted_2)
```

In [203]:
```
df1=pd.concat([knnPredictions,svcPredictions],axis=1)
```

In [204]:
```
df1.columns=[['knnPredictions','svcPredictions']]
```

In [205]:
```
df1
```

Out[205]:

	knnPredictions	svcPredictions
0	2	2
1	2	2
2	2	2
3	4	4
4	2	2
5	2	2
6	4	4
7	2	2
8	2	2
9	2	2
10	4	4
11	4	4
12	2	2
13	2	2
14	4	4
15	4	4
16	2	2
17	2	2
18	2	2
19	2	2
20	2	2
21	4	4
22	4	4
23	2	2
24	4	4
25	2	2
26	4	4
27	4	4
28	2	2
29	2	2
...	...	...
180	4	4
181	2	2
182	2	2
183	2	2
184	2	2
185	4	4
186	2	2
187	2	2
188	2	2
189	2	2

	knnPredictions	svcPredictions
190	2	2
191	2	2
192	2	2
193	4	4
194	2	2
195	4	4
196	4	4
197	4	4
198	4	4
199	2	2
200	2	2
201	2	2
202	2	2
203	2	2
204	4	4
205	4	4
206	4	4
207	2	2
208	2	2
209	2	2

210 rows × 2 columns

In [219]:
```
from sklearn.metrics import classification_report

print("classification_report for KNN")

print("..."*10)

print(classification_report(y_test, predicted_1))
```

```
classification_report for KNN
..............................
              precision    recall  f1-score   support

           2       0.96      0.99      0.98       137
           4       0.99      0.93      0.96        73

    accuracy                           0.97       210
   macro avg       0.98      0.96      0.97       210
weighted avg       0.97      0.97      0.97       210
```

In [220]:
```
from sklearn.metrics import classification_report

print("classification_report for SVC")

print("..."*10)

print(classification_report(y_test, predicted_2))
```

```
classification_report for SVC
..............................
              precision    recall  f1-score   support

           2       0.99      0.99      0.99       137
           4       0.97      0.97      0.97        73

    accuracy                           0.98       210
   macro avg       0.98      0.98      0.98       210
weighted avg       0.98      0.98      0.98       210
```

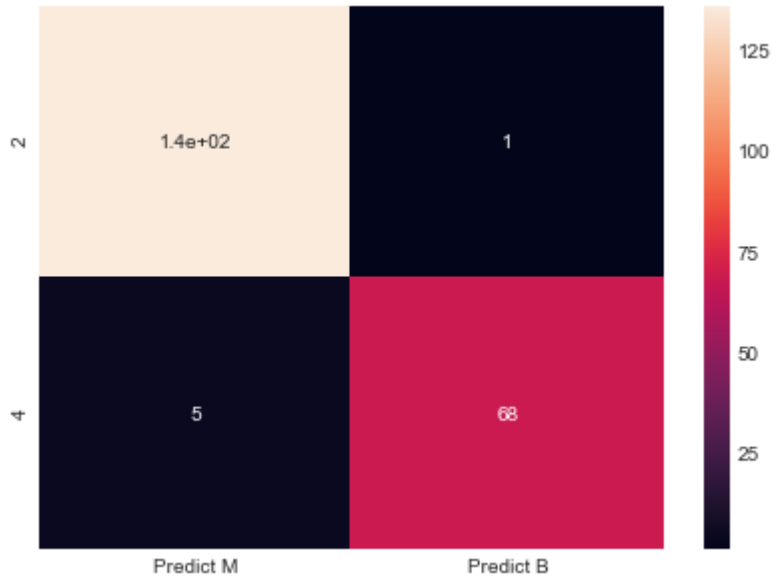
In [208]:
```
from sklearn import metrics

print("Confusion Matrix For KNeighborsClassifier")
cm=metrics.confusion_matrix(y_test, predicted_1, labels=[2, 4])

df_cm = pd.DataFrame(cm, index = [i for i in [2,4]],
                  columns = [i for i in ["Predict M","Predict B"]])
plt.figure(figsize = (7,5))
sns.heatmap(df_cm, annot=True)
```

```
Confusion Matrix For KNeighborsClassifier
```

Out[208]: <matplotlib.axes._subplots.AxesSubplot at 0x1a25cd1908>



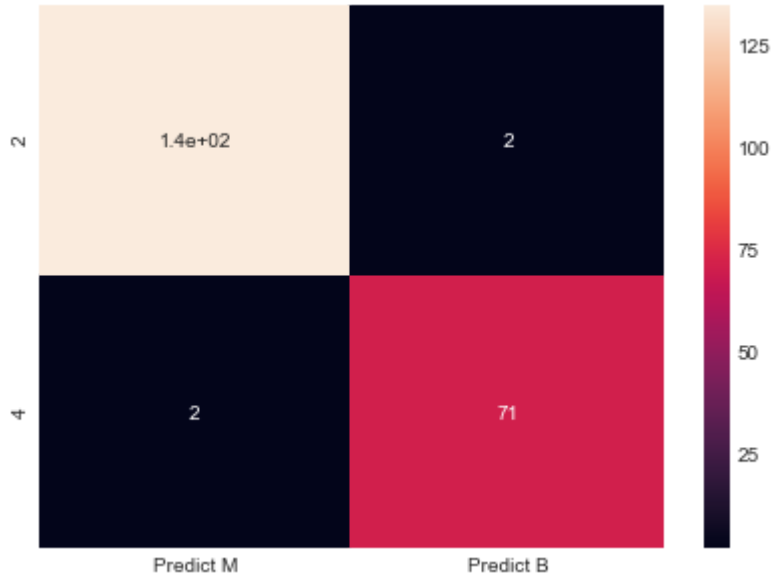
In [ ]:

In [209]:
```
print("Confusion Matrix For SupportVectorMachine")
cm=metrics.confusion_matrix(y_test, predicted_2, labels=[2, 4])

df_cm = pd.DataFrame(cm, index = [i for i in [2,4]],
                     columns = [i for i in ["Predict M","Predict B"]])
plt.figure(figsize = (7,5))
sns.heatmap(df_cm, annot=True)
```

```
Confusion Matrix For SupportVectorMachine
```

Out[209]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a269cb5f8>
```



In [ ]: