In [451]:

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

In [452]:

```
df = pd.read_csv('BreastCancerWc.csv',na_values='?')
```

In [453]:

```
df.columns =['Sample code number', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformit
df
```

Out[453]:

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Norr Nucle
0	1002945	5	4	4	5	7	10.0	3	
1	1015425	3	1	1	1	2	2.0	3	
2	1016277	6	8	8	1	3	4.0	3	
3	1017023	4	1	1	3	2	1.0	3	
4	1017122	8	10	10	8	7	10.0	9	
...	...	...	...	...	...	...	...	...	
693	776715	3	1	1	1	3	2.0	1	
694	841769	2	1	1	1	2	1.0	1	
695	888820	5	10	10	3	7	3.0	8	
696	897471	4	8	6	4	3	4.0	10	
697	897471	4	8	8	5	4	5.0	10	

698 rows × 11 columns

In [454]:

```
(df.astype(str)=='?').values.sum()
```

Out[454]:

```
0
```

In [455]:

```
df.isna().sum()
```

Out[455]:

```
Sample code number             0
Clump Thickness                0
Uniformity of Cell Size        0
Uniformity of 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 [456]:

```
df.shape
```

Out[456]:

```
(698, 11)
```

In [457]:

```
for i in df.columns:
    mean_value=df[i].mean()
    df[i]=df[i].fillna(mean_value)
```

In [458]:

```
df.isna().sum()
```

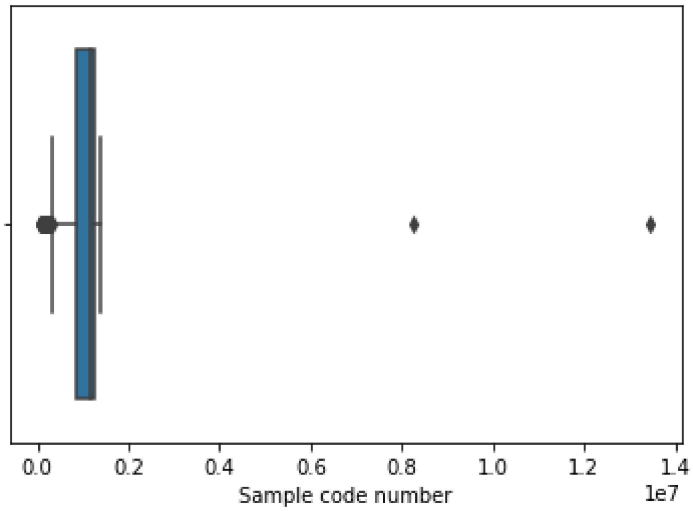
Out[458]:

```
Sample code number             0
Clump Thickness                0
Uniformity of Cell Size        0
Uniformity of Cell Shape       0
Marginal Adhesion              0
Single Epithelial Cell Size    0
Bare Nuclei                    0
Bland Chromatin                0
Normal Nucleoli                0
Mitoses                        0
Class                          0
dtype: int64
```

In [459]:

```
x=sns.boxplot(df['Sample code number'])
```

```
c:\users\adwait\appdata\local\programs\python\python38\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```



In [460]:

```
percentile25=df['Sample code number'].quantile(0.25)
percentile75=df['Sample code number'].quantile(0.75)
```

In [461]:

```
iqr = percentile75 - percentile25
iqr
```

Out[461]:

```
368095.75
```

In [462]:

```
upper_limit = percentile75 + 1.5 * iqr
lower_limit = percentile25 - 1.5 * iqr
```

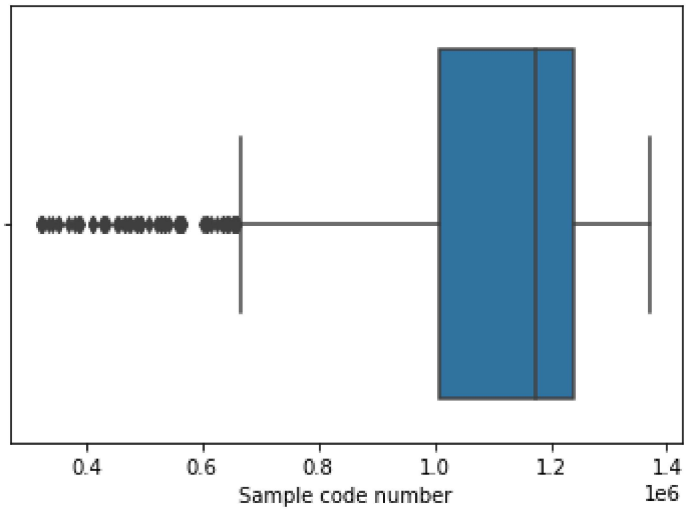
In [463]:

```
new_df=df[(df['Sample code number'] > lower_limit) & (df['Sample code number'] < upper_limi
```

In [464]:

```
x=sns.boxplot(new_df['Sample code number'])
```

```
c:\users\adwait\appdata\local\programs\python\python38\lib\site-packages\sea
born\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```



In [465]:

```
new_df.shape
```

Out[465]:

```
(675, 11)
```

In [466]:

```
df_norm = (new_df - new_df.min())/(new_df.max()-new_df.min())
```

In [467]:

```
df_norm.shape
```

Out[467]:

```
(675, 11)
```

In [468]:

```
df_norm.columns
```

Out[468]:

```
Index(['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'],
      dtype='object')
```

In [469]:

```
X = df_norm[['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']]
Y = df_norm['Class']
```

In [470]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,random_state=25)
```

In [471]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
```

In [472]:

```
model1 = LogisticRegression()
Y_train
```

Out[472]:

```
488    1.0
308    0.0
27     0.0
415    1.0
388    0.0
      ...
333    1.0
143    0.0
492    1.0
334    0.0
132    0.0
Name: Class, Length: 506, dtype: float64
```

In [473]:

```
model1.fit(X_train,Y_train)
```

Out[473]:

```
LogisticRegression()
```

In [474]:

```
y_pred = model1.predict(X_test)
```

In [475]:

```
y_pred
```

Out[475]:

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

In [476]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,y_pred)
```

Out[476]:

```
array([[115,   3],
       [  3,  48]], dtype=int64)
```

In [477]:

```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,y_pred)
```

Out[477]:

```
0.9644970414201184
```

In [ ]:

```

```