Importing Libraries

In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing Dataset --- Downloaded from UCI's Machine Learning Reprository

In [2]:
```python
df = pd.read_csv('cancer.csv')
df.head()
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280

5 rows × 33 columns

Overview :

We have to use 30 different columns and we have to predict the Stage of Breast Cancer M (Malignant) and B (Bengin)

Exploring and Cleaning Data

In [3]:
```python
df.columns
```

Out[3]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

By Exploring Data --- we saw some irrelevant columns which doesnt make any effect on tumor

- Id
- Unnamed32

In [4]:
```python
col = ['Unnamed: 32','id','diagnosis']
y = df.diagnosis
X = df.drop(col,axis=1)
```

In [5]:
```python
X.head()
```

Out[5]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

5 rows × 30 columns

In [6]:
```python
y.head()
```

Out[6]:
```
0    M
1    M
2    M
3    M
4    M
Name: diagnosis, dtype: object
```

In [7]:
```python
X.isnull().sum()
```

Out[7]:
```
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

There is No Missing column in dataset

In [8]:
```python
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   radius_mean              569 non-null    float64
 1   texture_mean             569 non-null    float64
 2   perimeter_mean           569 non-null    float64
 3   area_mean                569 non-null    float64
 4   smoothness_mean          569 non-null    float64
 5   compactness_mean         569 non-null    float64
 6   concavity_mean           569 non-null    float64
 7   concave points_mean      569 non-null    float64
 8   symmetry_mean            569 non-null    float64
 9   fractal_dimension_mean   569 non-null    float64
 10  radius_se                569 non-null    float64
 11  texture_se               569 non-null    float64
 12  perimeter_se             569 non-null    float64
 13  area_se                  569 non-null    float64
 14  smoothness_se            569 non-null    float64
 15  compactness_se           569 non-null    float64
 16  concavity_se             569 non-null    float64
 17  concave points_se        569 non-null    float64
 18  symmetry_se              569 non-null    float64
 19  fractal_dimension_se     569 non-null    float64
 20  radius_worst             569 non-null    float64
 21  texture_worst            569 non-null    float64
 22  perimeter_worst          569 non-null    float64
 23  area_worst               569 non-null    float64
 24  smoothness_worst         569 non-null    float64
 25  compactness_worst        569 non-null    float64
 26  concavity_worst          569 non-null    float64
 27  concave points_worst     569 non-null    float64
 28  symmetry_worst           569 non-null    float64
 29  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

All the values are real-valued (No categorical or object type variable ----) So , there is no need for Label_Encoding or One-hot-encoding

In [9]:
```python
X.describe()
```

Out[9]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800

8 rows × 30 columns

In [10]:
```python
X.corr()
```

Out[10]:

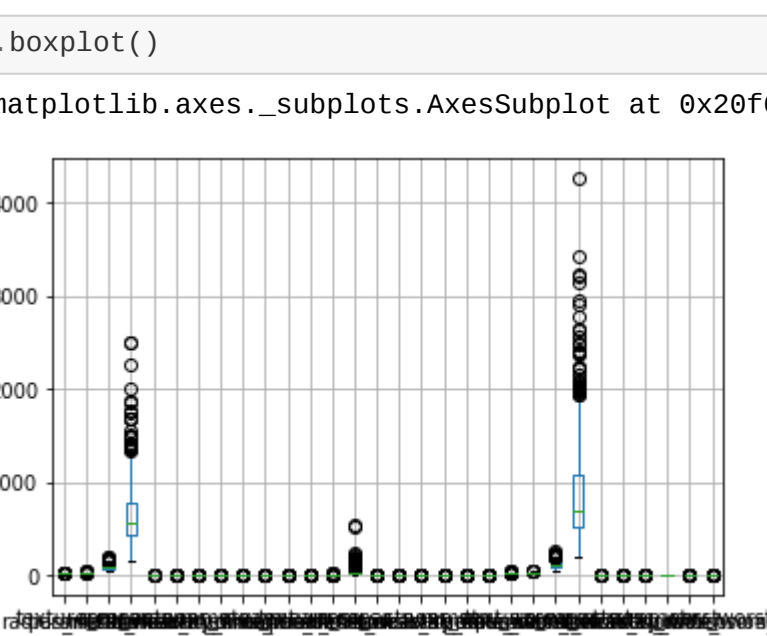
	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
radius_mean	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124
texture_mean	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702
perimeter_mean	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936
area_mean	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502
smoothness_mean	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123
compactness_mean	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000
concavity_mean	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121
concave points_mean	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135
symmetry_mean	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641
fractal_dimension_mean	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369
radius_se	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473
texture_se	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205
perimeter_se	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905
area_se	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653
smoothness_se	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299
compactness_se	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722
concavity_se	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517
concave points_se	0.376169	0.163851	0.407217	0.372320	0.380676	0.642262
symmetry_se	-0.104321	0.009127	-0.081629	-0.072497	0.200774	0.229977
fractal_dimension_se	-0.042641	0.054458	-0.005523	-0.019887	0.283607	0.507318
radius_worst	0.969539	0.352573	0.969476	0.962746	0.213120	0.535315
texture_worst	0.297008	0.912045	0.303038	0.287489	0.036072	0.248133
perimeter_worst	0.965137	0.358040	0.970387	0.959120	0.238853	0.590210
area_worst	0.941082	0.343546	0.941550	0.959213	0.206718	0.509604
smoothness_worst	0.119616	0.077503	0.150549	0.123523	0.805324	0.565541
compactness_worst	0.413463	0.277830	0.455774	0.390410	0.472468	0.865809
concavity_worst	0.526911	0.301025	0.563879	0.512606	0.434926	0.816275
concave points_worst	0.744214	0.295316	0.771241	0.722017	0.503053	0.815573
symmetry_worst	0.163953	0.105008	0.189115	0.143570	0.394309	0.510223
fractal_dimension_worst	0.007066	0.119205	0.051019	0.003738	0.499316	0.687382

30 rows × 30 columns

Data Vizualization :

In [11]:
```python
import seaborn as sns
sns.countplot(y,label='Count')
```

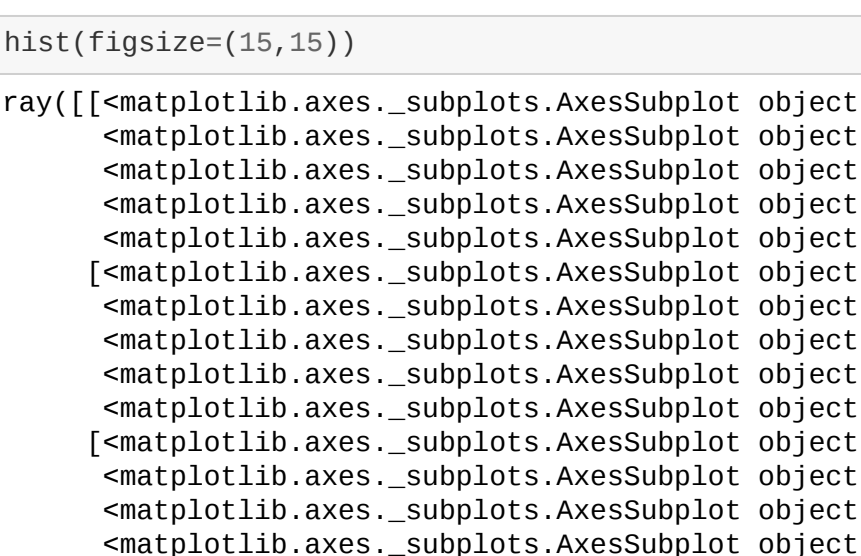
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x20f6b929188>



From this graph we can see that : There is a more number of Bengin stage of cancer

In [12]:
```python
X.boxplot()
```

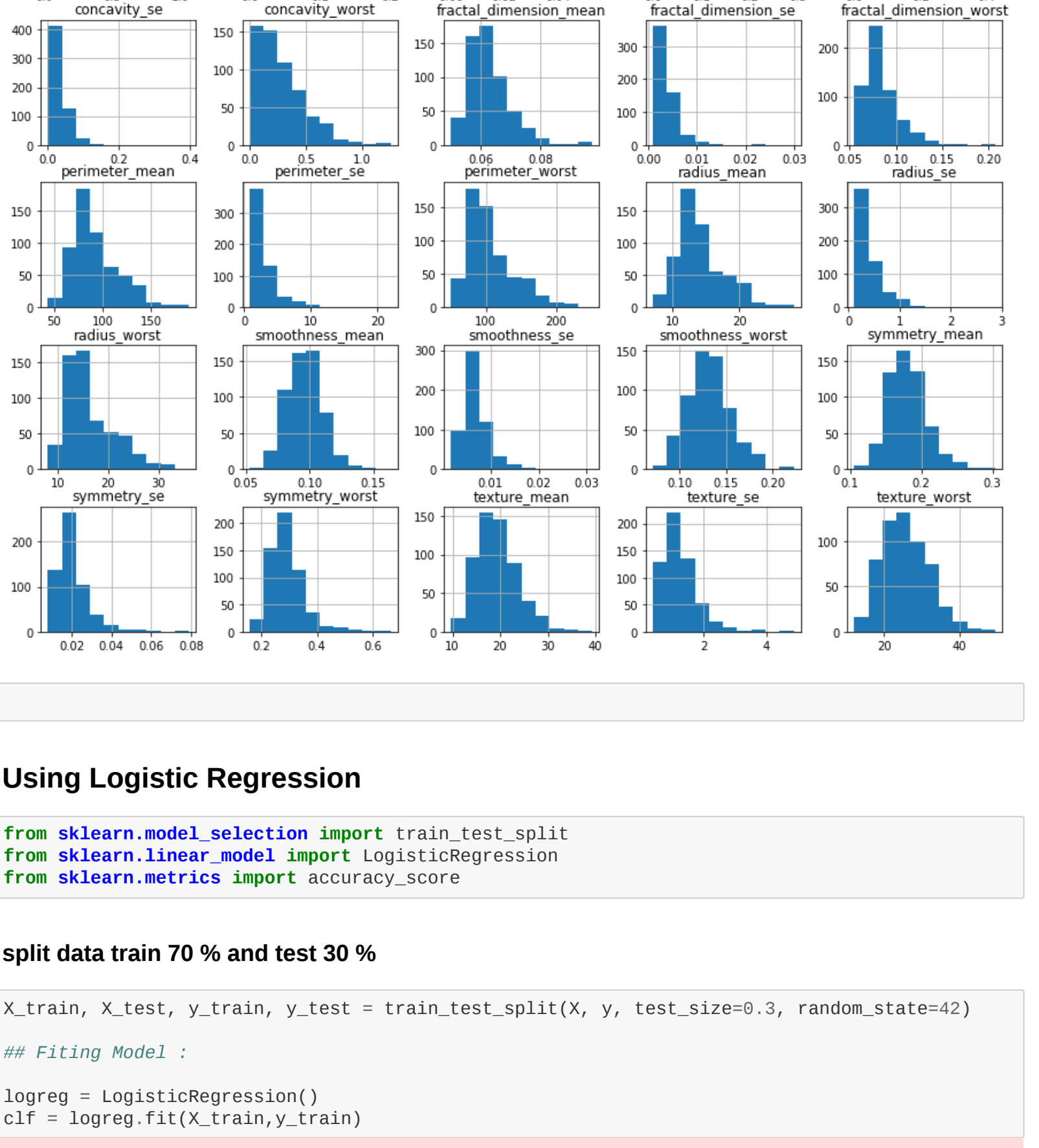
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x20f6c0c9ac8>



In [ ]:
```python
# from pandas.plotting import scatter_matrix
# scatter_matrix(df, alpha = 1, figsize =(15, 15),diagonal='hist')
```

In [13]:
```python
X.hist(figsize=(15,15))
```

Out[13]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6A876E88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C034748>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C468F48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C4A70C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C4DE1C8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C5182C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C5503C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C5884C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C593988>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C5CAB48>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C633808>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C66A8C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C6A4A08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C6DCB48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C716C48>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C74ED48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C785DC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C7BEEC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C7F6FC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C837148>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C86F248>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C8A8308>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C8DC3C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C9154C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C94F5C8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C985708>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C9C0808>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6C9F88C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6CA31A08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020F6CA67B08>]],
      dtype=object)
```



In [ ]:

Using Logistic Regression

In [14]:
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

split data train 70 % and test 30 %

In [15]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

## Fiting Model :

logreg = LogisticRegression()
clf = logreg.fit(X_train,y_train)
```

```
F:\CONDA\lib\site-packages\sklearn\linear_model\_logistic.py:940: ConvergenceWarning: lbfgs f
ailed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [16]:
```python
X_train.shape
```

Out[16]: (398, 30)

In [17]:
```python
X_test.shape
```

Out[17]: (171, 30)

In [18]:
```python
y_train.shape
```

Out[18]: (398,)

In [19]:
```python
y_test.shape
```

Out[19]: (171,)

Predict ---

In [20]:
```python
pred = logreg.predict(X_test)
```

Checking Accuracy of Model

In [21]:
```python
acc = accuracy_score(y_test, pred)
acc
```

Out[21]: 0.9707602339181286

Conclusion :

On visualizing the Data , We droped the irrelevant columns --- After applying Classification Algorithm Tht test set Accuracy is 0.97076 (97.08%)

In [ ]: