EXPERIMENT-2

AIM:

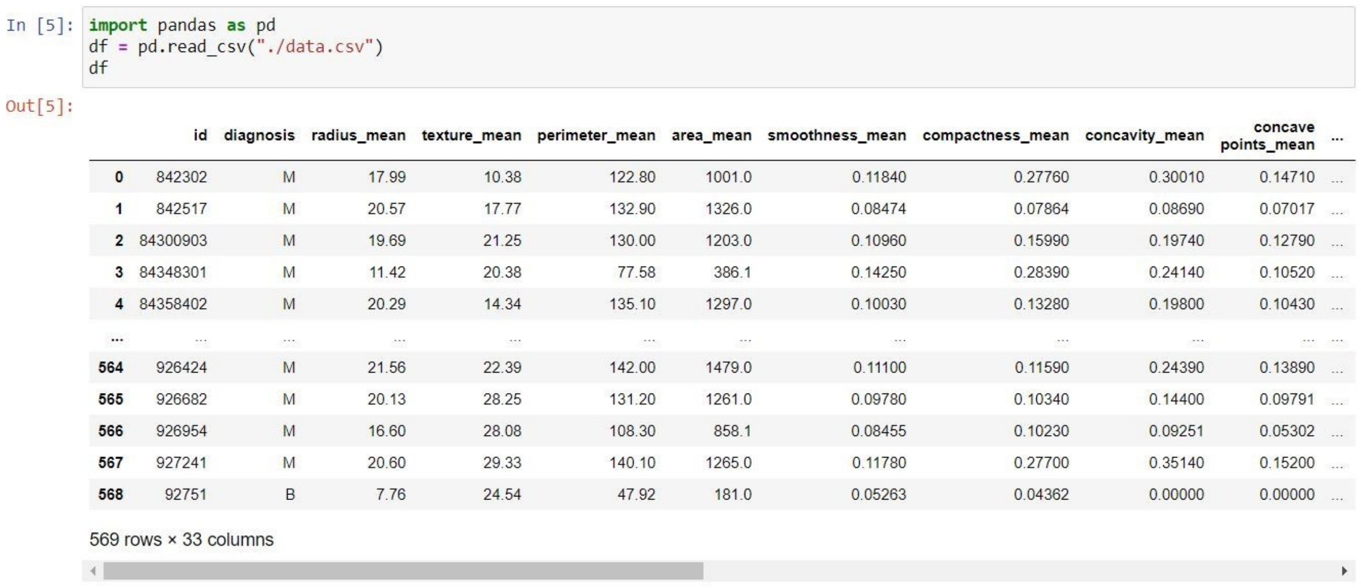
Study and implement the Naive Bayes learner on a breast cancer dataset

ALGORITHM:

1. Convert the data set into a frequency table
2. Create Likelihood table by finding the probabilities.
3. Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction

PROGRAM CODE SNIPPET:

LOADINGDATA SET:



```
In [5]: import pandas as pd
        df = pd.read_csv("./data.csv")
        df
```

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...
...	...	...	...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...

569 rows × 33 columns

PREPROCESSING:

In [5]:
```
#to read the last end of data
df.tail()
```

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	te
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	

5 rows × 33 columns

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [7]:
```
df.shape
```

Out[7]:
```
(569, 33)
```

In [8]:
```
#print all the columns of dataset
df.columns.values
```

Out[8]:
```
array(['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)
```

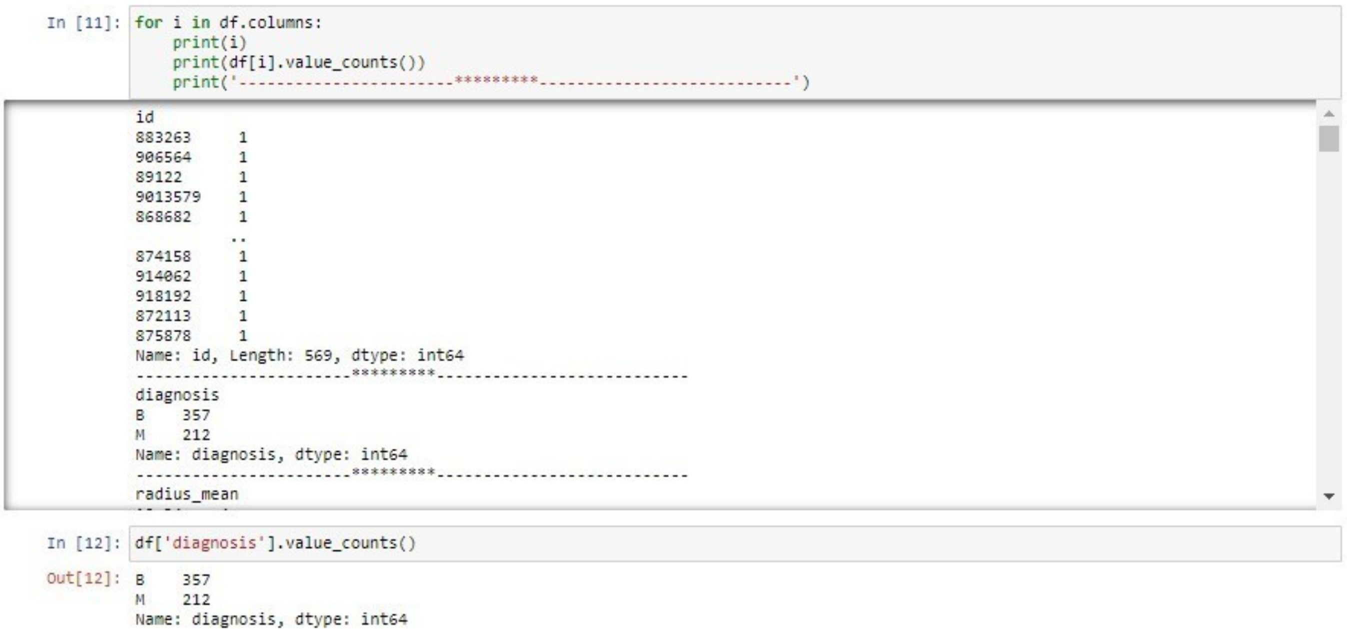
```
In [9]: df.corr()
```

Out[9]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conc points_m
id	1.000000	0.074626	0.099770	0.073159	0.096893	-0.012968	0.000096	0.050080	0.04
radius_mean	0.074626	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.82
texture_mean	0.099770	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.29
perimeter_mean	0.073159	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.85
area_mean	0.096893	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.82
smoothness_mean	-0.012968	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.55
compactness_mean	0.000096	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.83
concavity_mean	0.050080	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.92
concave points_mean	0.044158	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.00
symmetry_mean	-0.022114	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.46
fractal_dimension_mean	-0.052511	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.16
radius_se	0.143048	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.69
texture_se	-0.007526	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.02
perimeter_se	0.137331	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.71
area_se	0.177742	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653	0.617427	0.69
smoothness_se	0.096781	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564	0.02
compactness_se	0.033961	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722	0.670279	0.49
concavity_se	0.055239	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270	0.43
concave points_se	0.078768	0.376169	0.163851	0.407217	0.372320	0.380676	0.642262	0.683260	0.61
symmetry_se	-0.017306	-0.104321	0.009127	-0.081629	-0.072497	0.200774	0.229977	0.178009	0.09
fractal_dimension_se	0.025725	-0.042641	0.054458	-0.005523	-0.019887	0.283607	0.507318	0.449301	0.25
radius_worst	0.082405	0.969539	0.352573	0.969476	0.962746	0.213120	0.535315	0.688236	0.83
texture_worst	0.064720	0.297008	0.912045	0.303038	0.287489	0.036072	0.248133	0.299879	0.29
perimeter_worst	0.079986	0.965137	0.358040	0.970387	0.959120	0.238853	0.590210	0.729565	0.85

```
In [10]: #check for the null value
         df.isnull().sum()
```

```
Out[10]: id                          0
         diagnosis                   0
         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
         Unnamed: 32               569
         dtype: int64
```

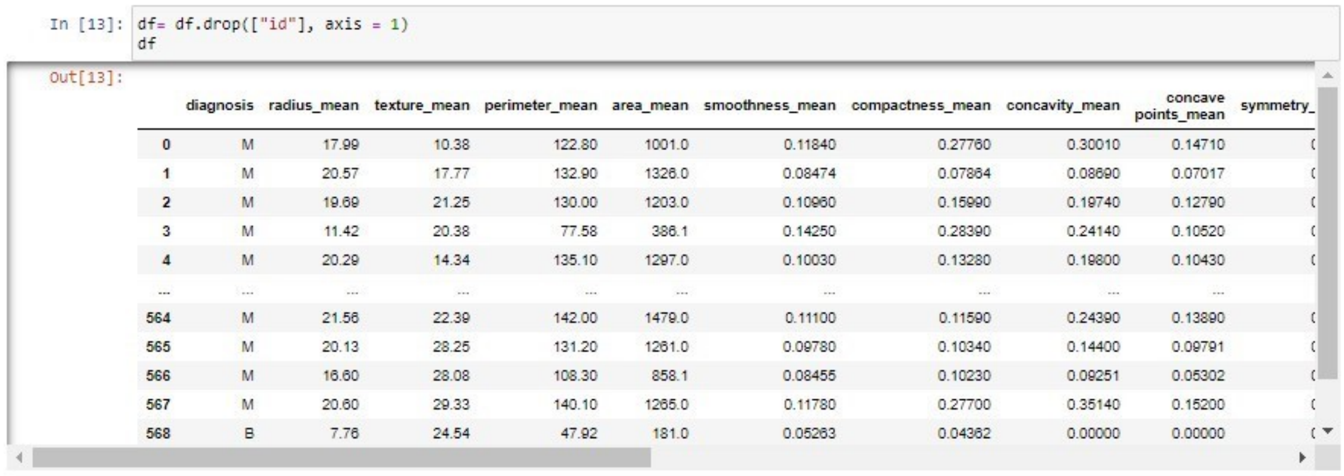


In [11]:
```
for i in df.columns:
    print(i)
    print(df[i].value_counts())
    print('----------------------*********-------------------------')
```

```
id
883263     1
906564     1
89122      1
9013579    1
868682     1
          ..
874158     1
914062     1
918192     1
872113     1
875878     1
Name: id, Length: 569, dtype: int64
----------------------*********-------------------------
diagnosis
B    357
M    212
Name: diagnosis, dtype: int64
----------------------*********-------------------------
radius_mean
```

In [12]:
```
df['diagnosis'].value_counts()
```

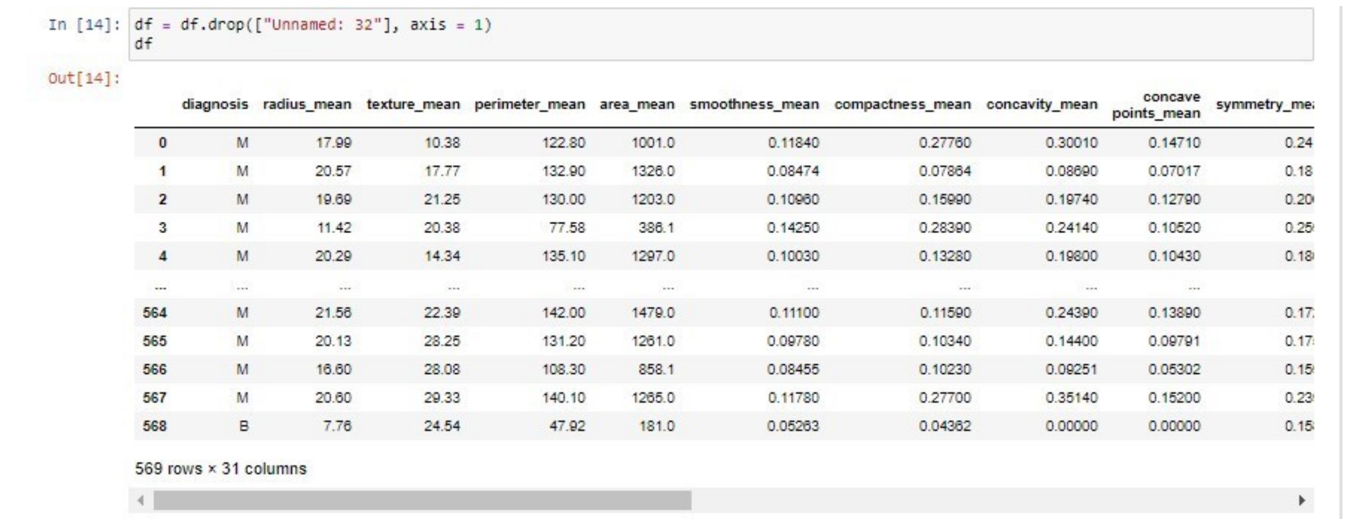
Out[12]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```



In [13]:
```
df= df.drop(["id"], axis = 1)
df
```

Out[13]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0
...	...	...	...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0
565	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0
566	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0
567	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0
568	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0



In [14]:
```
df = df.drop(["Unnamed: 32"], axis = 1)
df
```

Out[14]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_me
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.24
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.18
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.20
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.25
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.18
...	...	...	...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.17
565	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.17
566	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.15
567	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.23
568	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.15

569 rows × 31 columns

VISUALIZATION:

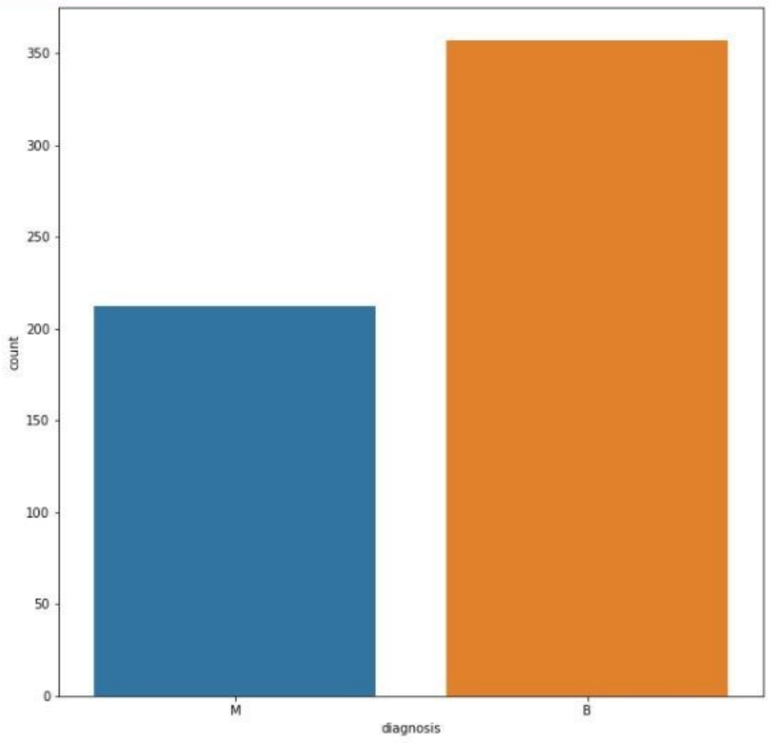
```
In [15]: import matplotlib.pyplot as plt
         import seaborn as sns
```

```
In [16]: benign, malignant=df['diagnosis'].value_counts()
         print("No of Benign cell", benign)
         print("No of malignant cell", malignant)
```

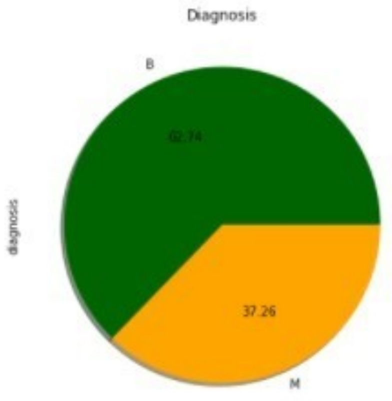
```
No of Benign cell 357
No of malignant cell 212
```

```
In [19]: plt.figure(figsize=(10,10))
         sns.countplot(df['diagnosis'])
         plt.show()
```

```
C:\Users\Is_dhillon\anaconda3\lib\site-packages\seaborn\_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 `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```

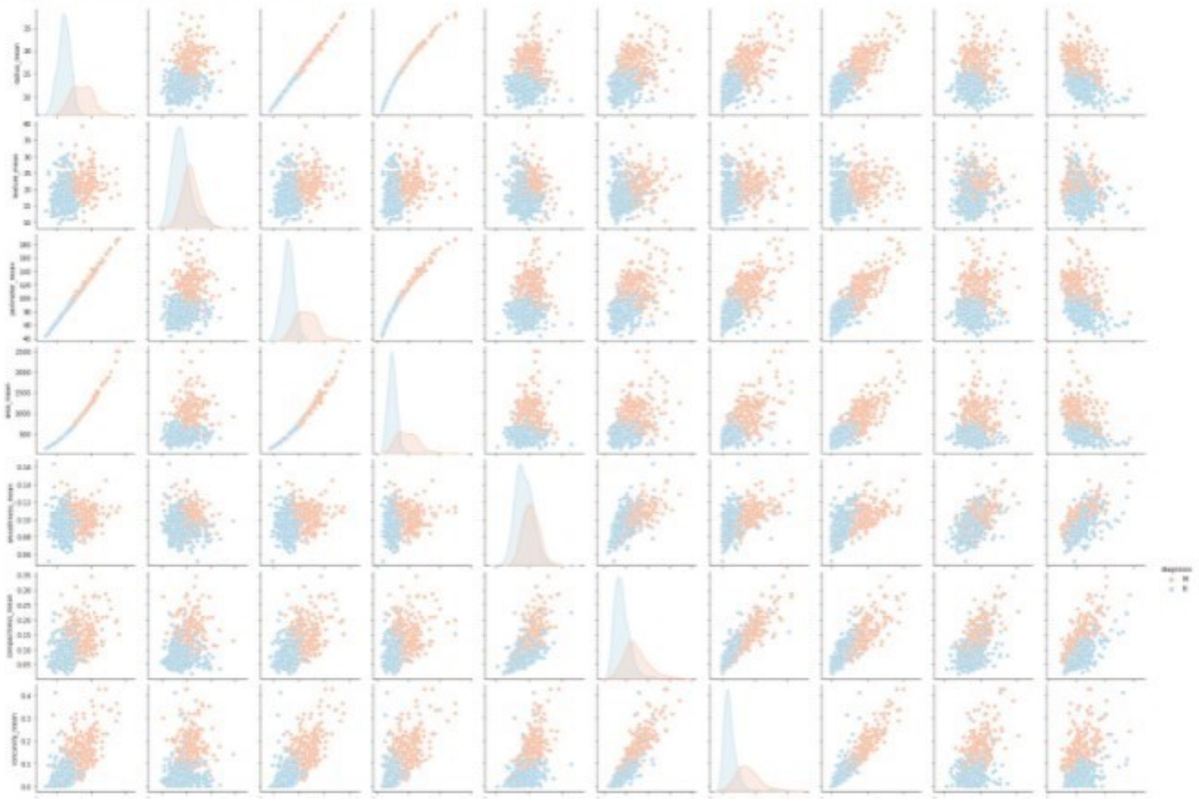


```
In [18]: print("% of Benign cell is ", benign*100/len(df))
         print("% of Malignant cell is ", malignant*100/len(df))

         % of Benign cell is  62.74165202108963
         % of Malignant cell is  37.25834797891037

In [19]: df.diagnosis.value_counts().plot(kind='pie',shadow=True,colors=('darkgreen','orange'),autopct='%.2f',figsize=(8,6))
         plt.title('Diagnosis')
         plt.show()
```



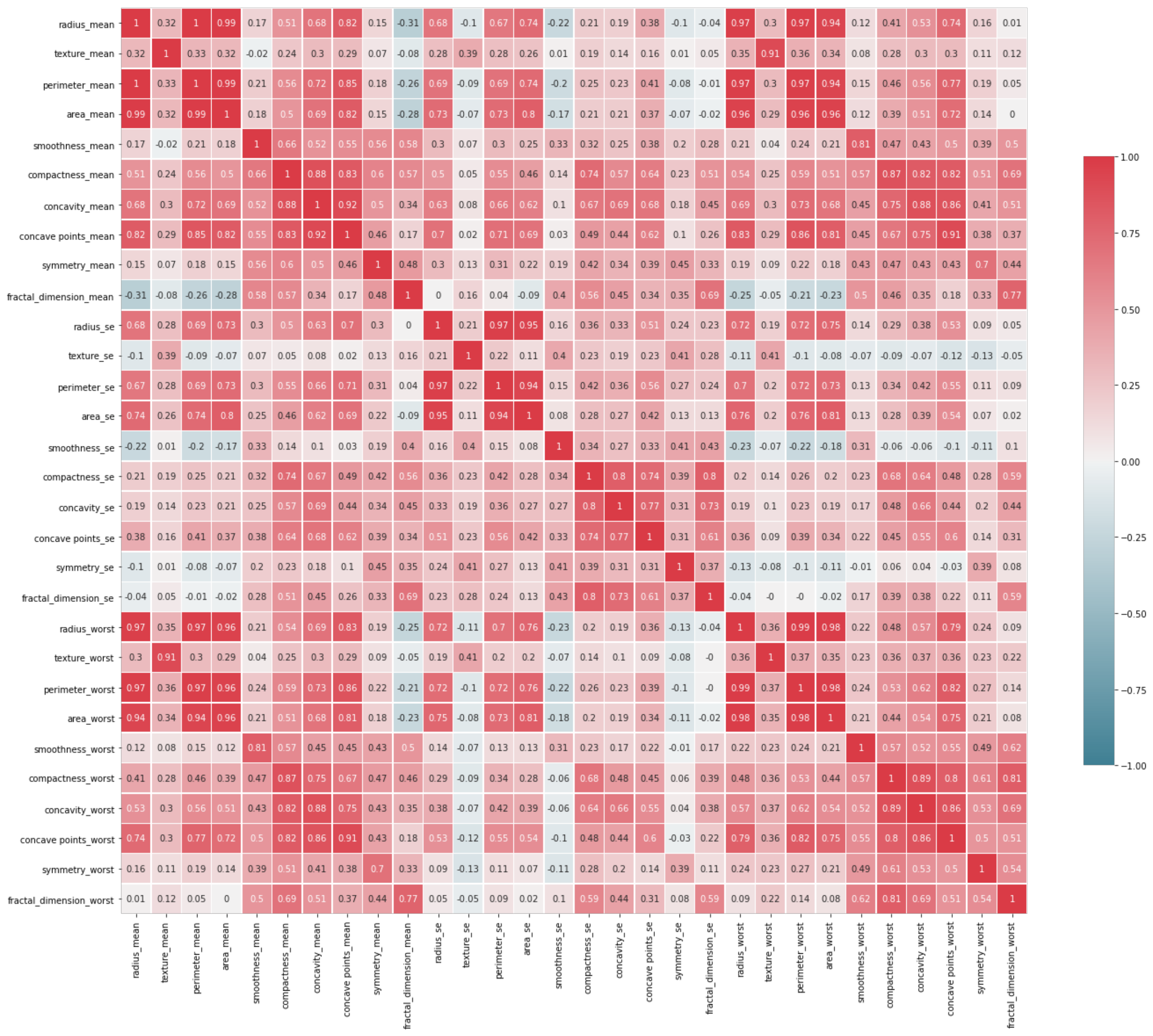
Pairplot helps to plot among the most useful feature

```
In [20]: cols=['diagnosis','radius_mean', 'texture_mean', 'perimeter_mean',
                'area_mean','smoothness_mean', 'compactness_mean', 'concavity_mean',
                'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean']
         plt.figure(figsize=(10,10))
         sns.pairplot(data=df[cols],hue='diagnosis', palette='RdBu')

Out[20]: <seaborn.axisgrid.PairGrid at 0x276b14608b0>

         <Figure size 720x720 with 0 Axes>
```

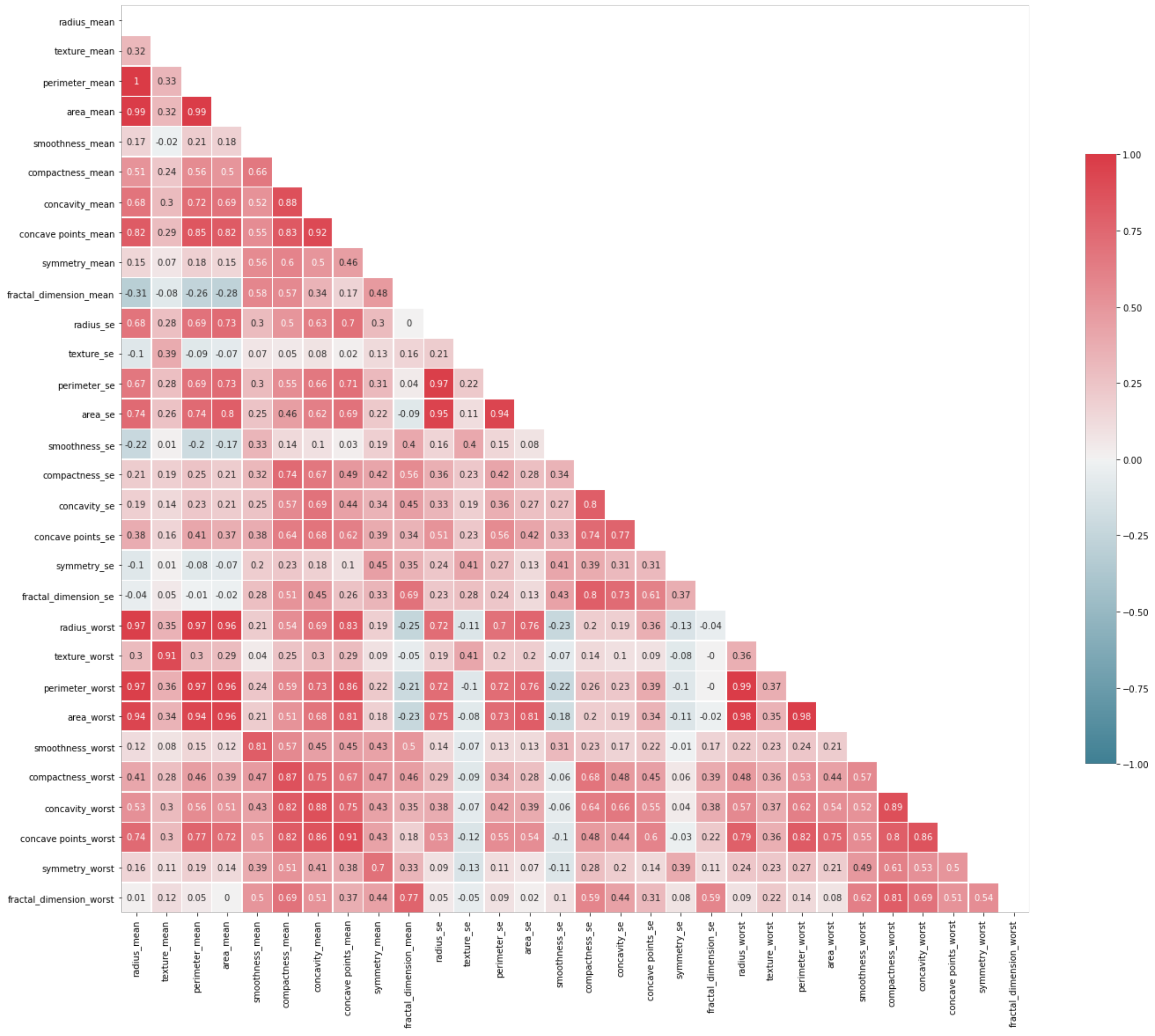


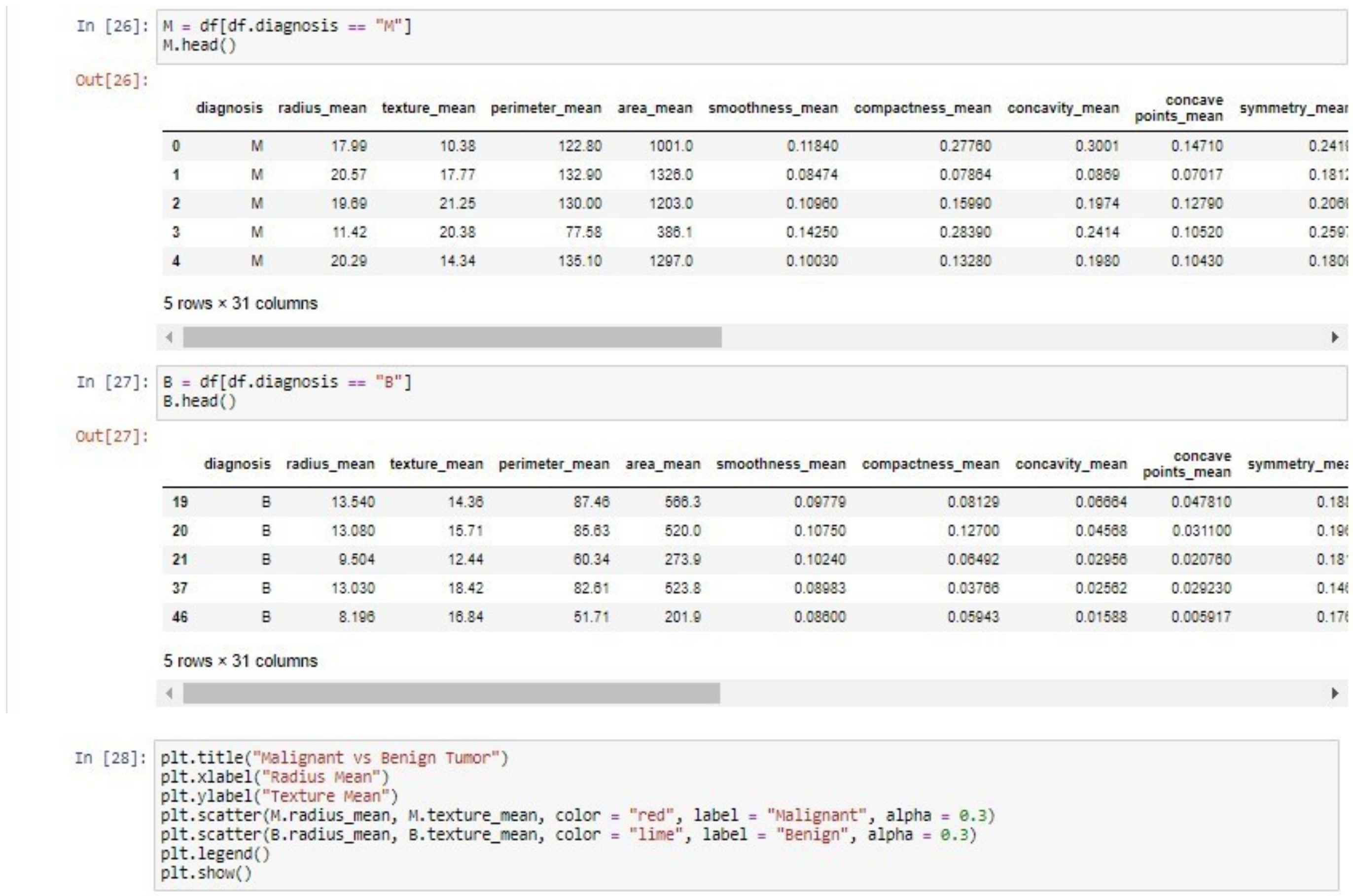
```
In [23]: import numpy as np
```

```
In [24]: #generate the corelation matrix
         corr=df.corr().round(2)
         #mask for the upper triangle
         mask=np.zeros_like(corr, dtype=np.bool)
         mask[np.triu_indices_from(mask)]
         # Set figure size
         f, ax = plt.subplots(figsize=(20, 20))

         #define custom colormap
         cmap=sns.diverging_palette(220,10, as_cmap=True)

         #draw the heatmap
         sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
                     square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True)

         plt.tight_layout()
```



In [25]:
```python
# Generate and visualize the correlation matrix
corr = df.corr().round(2)

# Mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set figure size
f, ax = plt.subplots(figsize=(20, 20))

# Define custom colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True)

plt.tight_layout()
```





```
In [26]: M = df[df.diagnosis == "M"]
         M.head()
```

Out[26]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mea
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.241
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.181
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.206
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.259
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.180

5 rows × 31 columns

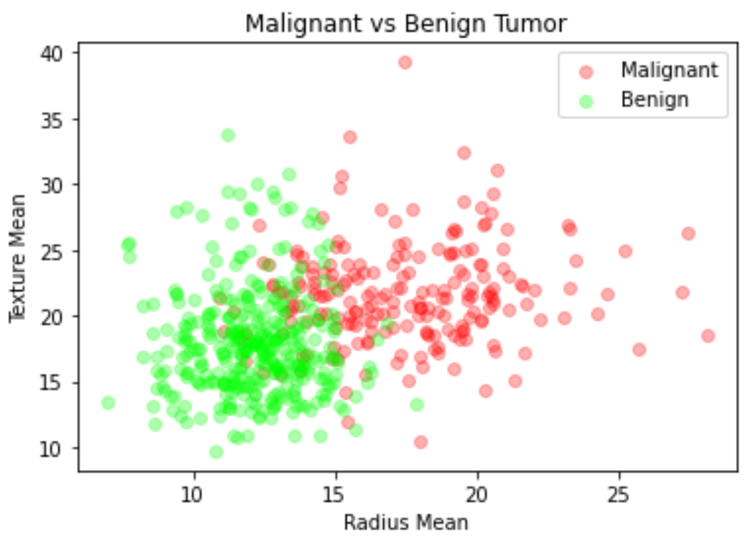
```
In [27]: B = df[df.diagnosis == "B"]
         B.head()
```

Out[27]:

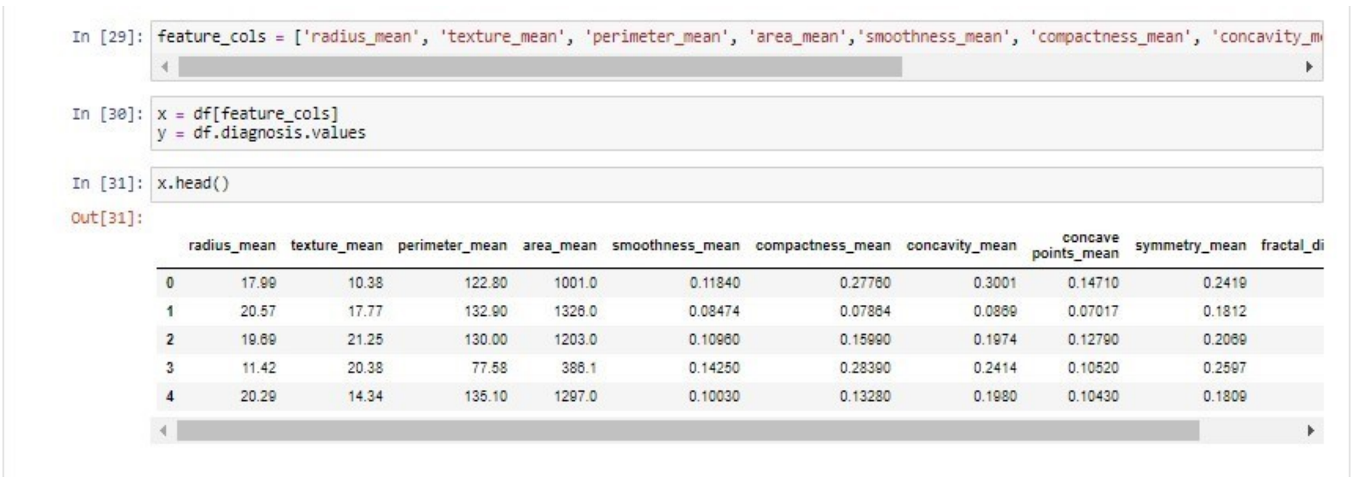
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mea
19	B	13.540	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.047810	0.18
20	B	13.080	15.71	85.63	520.0	0.10750	0.12700	0.04568	0.031100	0.19
21	B	9.504	12.44	60.34	273.9	0.10240	0.06492	0.02956	0.020760	0.18
37	B	13.030	18.42	82.61	523.8	0.08983	0.03766	0.02562	0.029230	0.14
46	B	8.196	16.84	51.71	201.9	0.08600	0.05943	0.01588	0.005917	0.17

5 rows × 31 columns

```
In [28]: plt.title("Malignant vs Benign Tumor")
         plt.xlabel("Radius Mean")
         plt.ylabel("Texture Mean")
         plt.scatter(M.radius_mean, M.texture_mean, color = "red", label = "Malignant", alpha = 0.3)
         plt.scatter(B.radius_mean, B.texture_mean, color = "lime", label = "Benign", alpha = 0.3)
         plt.legend()
         plt.show()
```



ML ALGORITHM IMPLEMENTATION:



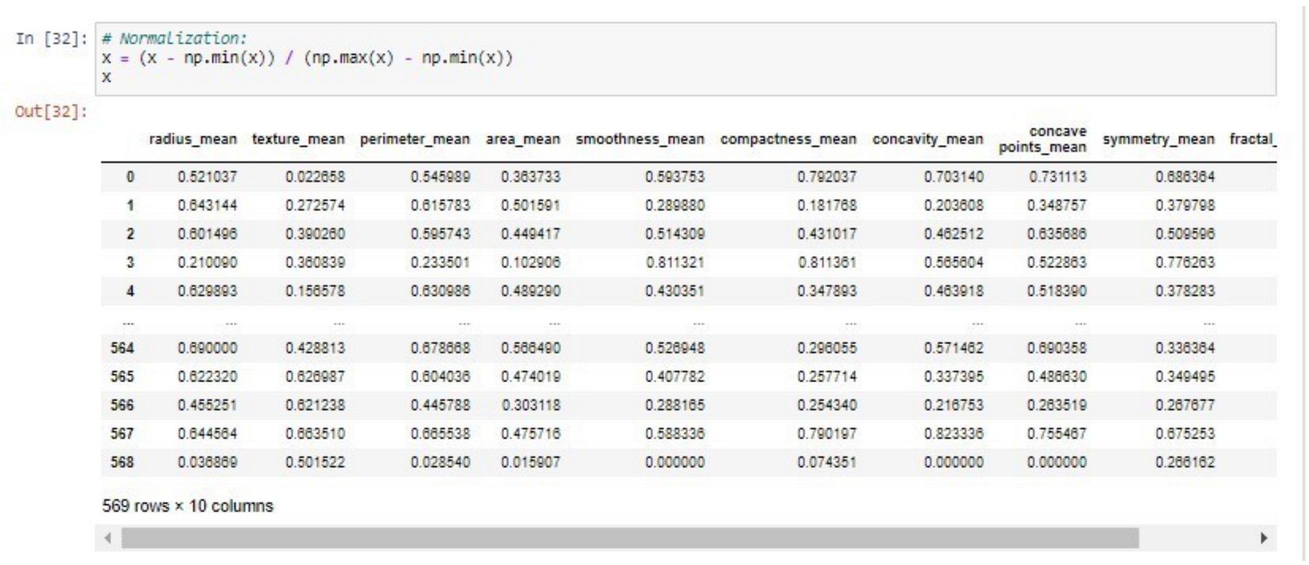
```
In [29]: feature_cols = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean','smoothness_mean', 'compactness_mean', 'concavity_m
```

```
In [30]: x = df[feature_cols]
         y = df.diagnosis.values
```

```
In [31]: x.head()
```

Out[31]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_di
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	

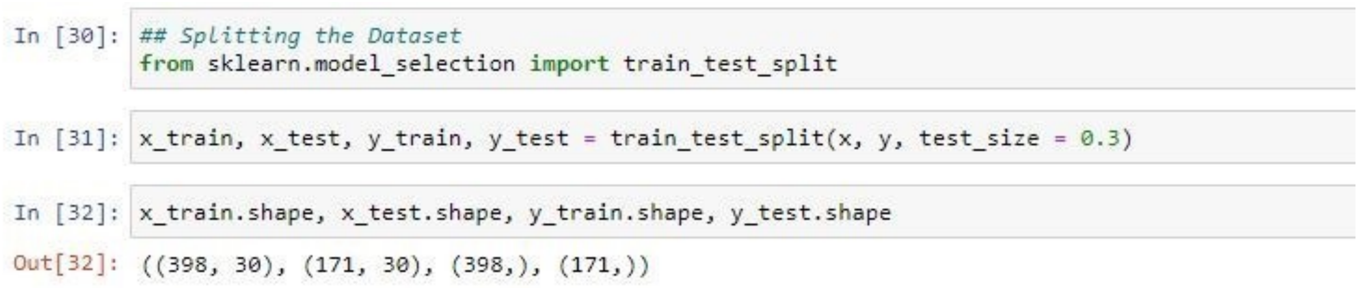


```
In [32]: # Normalization:
         x = (x - np.min(x)) / (np.max(x) - np.min(x))
         x
```

Out[32]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_
0	0.521037	0.022658	0.545989	0.363733	0.593753	0.792037	0.703140	0.731113	0.686364	
1	0.643144	0.272574	0.615783	0.501591	0.289880	0.181768	0.203608	0.348757	0.379798	
2	0.601496	0.390260	0.595743	0.449417	0.514309	0.431017	0.462512	0.635686	0.509596	
3	0.210090	0.360839	0.233501	0.102906	0.811321	0.811361	0.565604	0.522863	0.776263	
4	0.629893	0.156578	0.630986	0.489290	0.430351	0.347893	0.463918	0.518390	0.378283	
...	...	...	...	...	...	...	...	...	...	
564	0.690000	0.428813	0.678668	0.566490	0.526948	0.296055	0.571462	0.690358	0.336364	
565	0.622320	0.626987	0.604036	0.474019	0.407782	0.257714	0.337395	0.486630	0.349495	
566	0.455251	0.621238	0.445788	0.303118	0.288165	0.254340	0.216753	0.263519	0.267677	
567	0.644564	0.663510	0.665538	0.475716	0.588336	0.790197	0.823336	0.755467	0.675253	
568	0.036869	0.501522	0.028540	0.015907	0.000000	0.074351	0.000000	0.000000	0.266162	

569 rows × 10 columns



```
In [30]: ## Splitting the Dataset
         from sklearn.model_selection import train_test_split
```

```
In [31]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

```
In [32]: x_train.shape, x_test.shape, y_train.shape, y_test.shape

Out[32]: ((398, 30), (171, 30), (398,), (171,))
```

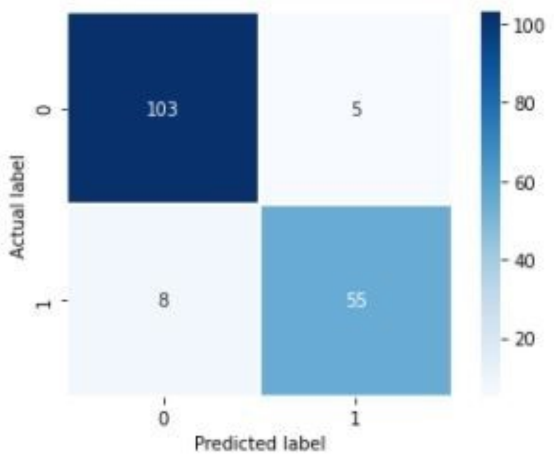
```
In [39]: ## Applying the Naive Bayes
         from sklearn.naive_bayes import GaussianNB
         nb = GaussianNB()
         nb.fit(x_train, y_train)

         print("Naive Bayes score: ",nb.score(x_test, y_test))

         Naive Bayes score:  0.9239766081871345
```

```
In [40]: from sklearn.model_selection import train_test_split
         from sklearn.metrics import classification_report, confusion_matrix
         from sklearn.tree import plot_tree
         y_pred = nb.predict(x_test)
         cm=confusion_matrix(y_test,y_pred)
         cm

Out[40]: array([[103,   5],
                [  8,  55]], dtype=int64)
```

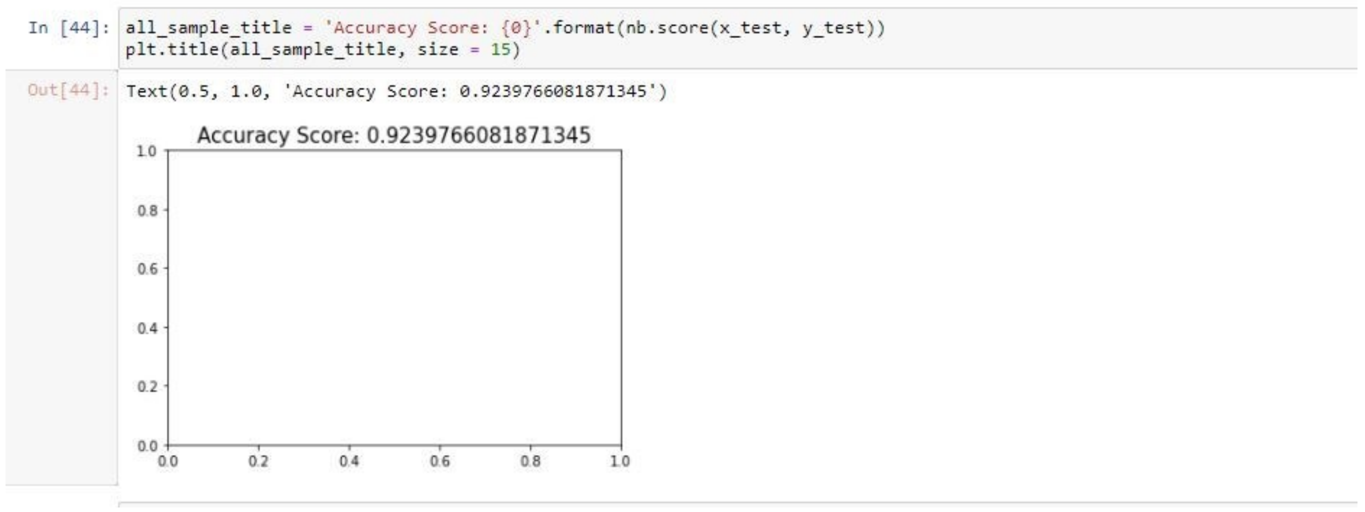
```
In [41]: import matplotlib.pyplot as plt
         import seaborn as sns
         pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

```
In [42]: plt.figure(figsize=(5,5))

Out[42]: <Figure size 360x360 with 0 Axes>

         <Figure size 360x360 with 0 Axes>
```

```
In [45]: sns.heatmap(data=cm,linewidths=1.0, annot=True,square = True,  cmap = 'Blues', fmt='g')
         plt.ylabel('Actual label')
         plt.xlabel('Predicted label')

Out[45]: Text(0.5, 15.0, 'Predicted label')
```



FINAL RESULT:



```
In [44]: all_sample_title = 'Accuracy Score: {0}'.format(nb.score(x_test, y_test))
         plt.title(all_sample_title, size = 15)

Out[44]: Text(0.5, 1.0, 'Accuracy Score: 0.9239766081871345')
```

GITHUB LINK:
https://github.com/tejpalsingh1999/Machine-Learning/tree/master/Exp%202