Avnish Singh jaswal

00113207218

CSE- 1

MACHINE LEARNING

LAB PROGRAM 1

EXPERIMENT-1

AIM:

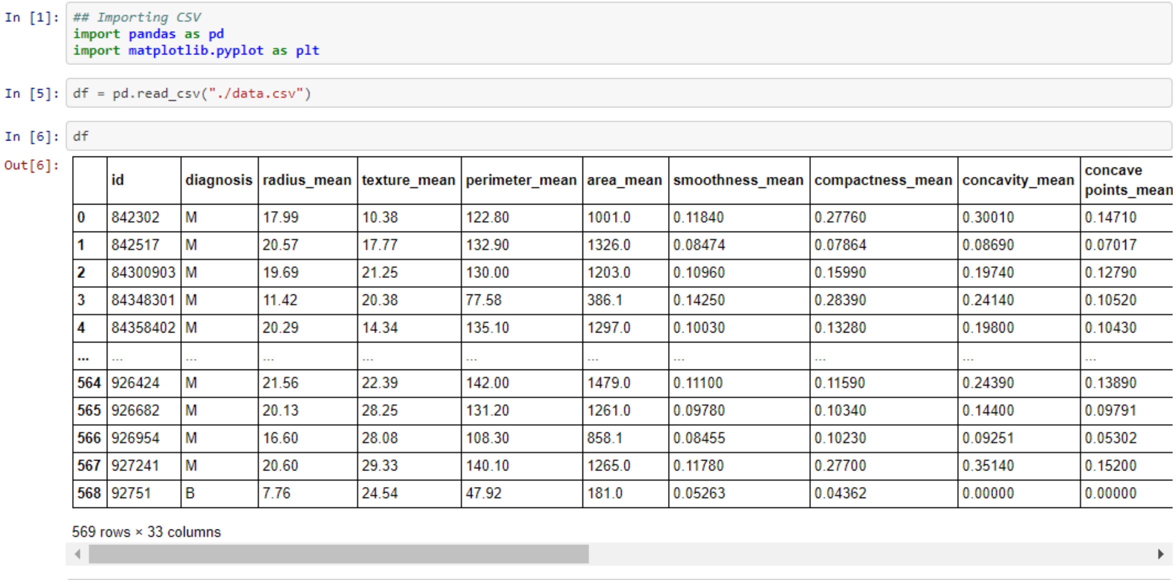
Study and implement the Naive Bayes learner on a breast cancer dataset

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:

LOADING DATA SET:



```
In [1]: ## Importing CSV
        import pandas as pd
        import matplotlib.pyplot as plt

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

In [6]: df
Out[6]:
```

	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

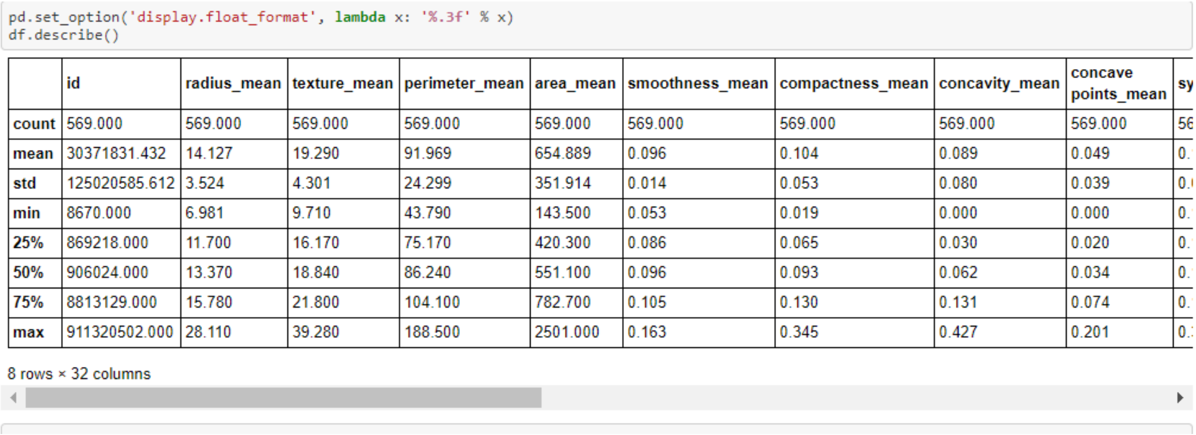
569 rows × 33 columns

PREPROCESSING:

```
"""## Analyzing and Cleaning Data"""

## Getting Info about Dataset
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
```



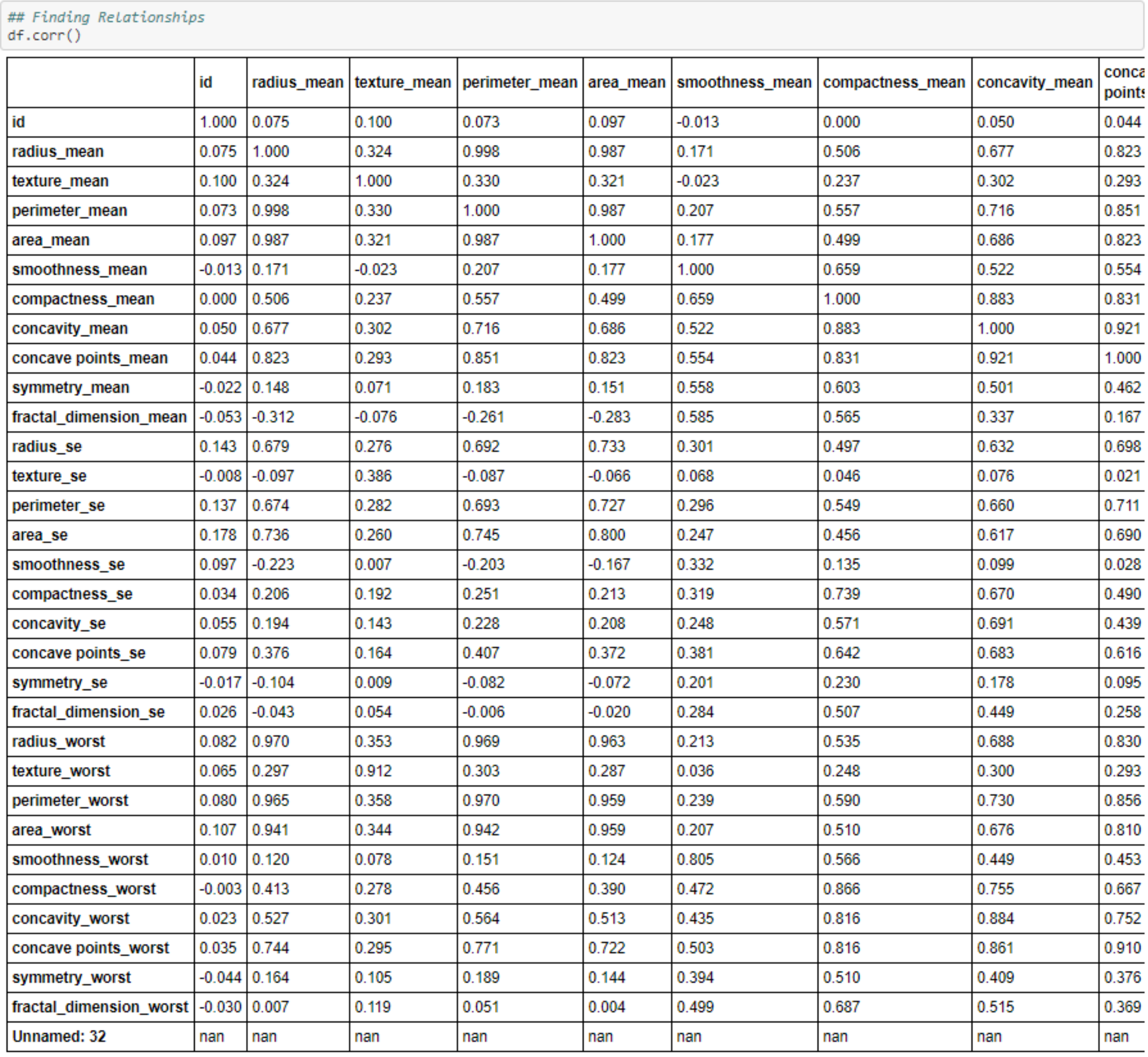
```
pd.set_option('display.float_format', lambda x: '%.3f' % x)
df.describe()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	sy
count	569.000	569.000	569.000	569.000	569.000	569.000	569.000	569.000	569.000	56
mean	30371831.432	14.127	19.290	91.969	654.889	0.096	0.104	0.089	0.049	0.
std	125020585.612	3.524	4.301	24.299	351.914	0.014	0.053	0.080	0.039	0.
min	8670.000	6.981	9.710	43.790	143.500	0.053	0.019	0.000	0.000	0.
25%	869218.000	11.700	16.170	75.170	420.300	0.086	0.065	0.030	0.020	0.
50%	906024.000	13.370	18.840	86.240	551.100	0.096	0.093	0.062	0.034	0.
75%	8813129.000	15.780	21.800	104.100	782.700	0.105	0.130	0.131	0.074	0.
max	911320502.000	28.110	39.280	188.500	2501.000	0.163	0.345	0.427	0.201	0.

8 rows × 32 columns

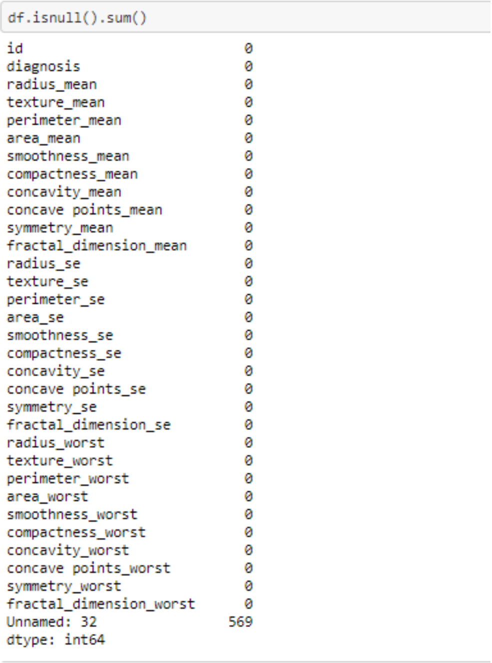
```
df.shape
```

```
(569, 33)
```



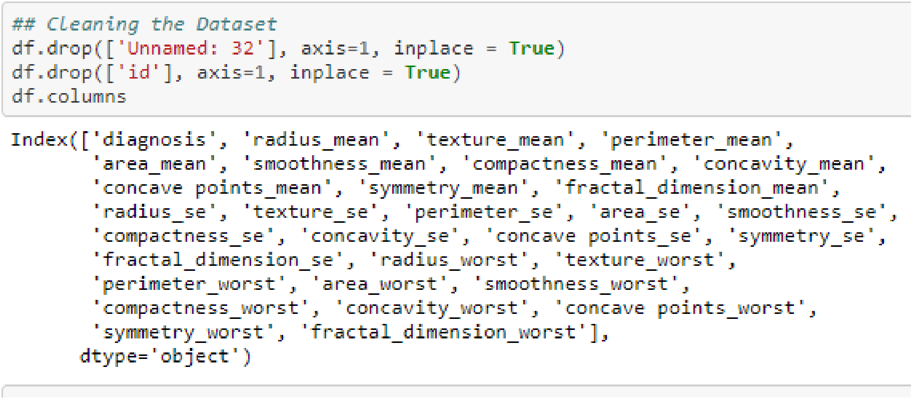
```
## Finding Relationships
df.corr()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conca points
id	1.000	0.075	0.100	0.073	0.097	-0.013	0.000	0.050	0.044
radius_mean	0.075	1.000	0.324	0.998	0.987	0.171	0.506	0.677	0.823
texture_mean	0.100	0.324	1.000	0.330	0.321	-0.023	0.237	0.302	0.293
perimeter_mean	0.073	0.998	0.330	1.000	0.987	0.207	0.557	0.716	0.851
area_mean	0.097	0.987	0.321	0.987	1.000	0.177	0.499	0.686	0.823
smoothness_mean	-0.013	0.171	-0.023	0.207	0.177	1.000	0.659	0.522	0.554
compactness_mean	0.000	0.506	0.237	0.557	0.499	0.659	1.000	0.883	0.831
concavity_mean	0.050	0.677	0.302	0.716	0.686	0.522	0.883	1.000	0.921
concave points_mean	0.044	0.823	0.293	0.851	0.823	0.554	0.831	0.921	1.000
symmetry_mean	-0.022	0.148	0.071	0.183	0.151	0.558	0.603	0.501	0.462
fractal_dimension_mean	-0.053	-0.312	-0.076	-0.261	-0.283	0.585	0.565	0.337	0.167
radius_se	0.143	0.679	0.276	0.692	0.733	0.301	0.497	0.632	0.698
texture_se	-0.008	-0.097	0.386	-0.087	-0.066	0.068	0.046	0.076	0.021
perimeter_se	0.137	0.674	0.282	0.693	0.727	0.296	0.549	0.660	0.711
area_se	0.178	0.736	0.260	0.745	0.800	0.247	0.456	0.617	0.690
smoothness_se	0.097	-0.223	0.007	-0.203	-0.167	0.332	0.135	0.099	0.028
compactness_se	0.034	0.206	0.192	0.251	0.213	0.319	0.739	0.670	0.490
concavity_se	0.055	0.194	0.143	0.228	0.208	0.248	0.571	0.691	0.439
concave points_se	0.079	0.376	0.164	0.407	0.372	0.381	0.642	0.683	0.616
symmetry_se	-0.017	-0.104	0.009	-0.082	-0.072	0.201	0.230	0.178	0.095
fractal_dimension_se	0.026	-0.043	0.054	-0.006	-0.020	0.284	0.507	0.449	0.258
radius_worst	0.082	0.970	0.353	0.969	0.963	0.213	0.535	0.688	0.830
texture_worst	0.065	0.297	0.912	0.303	0.287	0.036	0.248	0.300	0.293
perimeter_worst	0.080	0.965	0.358	0.970	0.959	0.239	0.590	0.730	0.856
area_worst	0.107	0.941	0.344	0.942	0.959	0.207	0.510	0.676	0.810
smoothness_worst	0.010	0.120	0.078	0.151	0.124	0.805	0.566	0.449	0.453
compactness_worst	-0.003	0.413	0.278	0.456	0.390	0.472	0.866	0.755	0.667
concavity_worst	0.023	0.527	0.301	0.564	0.513	0.435	0.816	0.884	0.752
concave points_worst	0.035	0.744	0.295	0.771	0.722	0.503	0.816	0.861	0.910
symmetry_worst	-0.044	0.164	0.105	0.189	0.144	0.394	0.510	0.409	0.376
fractal_dimension_worst	-0.030	0.007	0.119	0.051	0.004	0.499	0.687	0.515	0.369
Unnamed: 32	nan	nan	nan	nan	nan	nan	nan	nan	nan



```
df.isnull().sum()
```

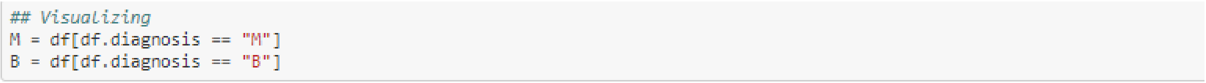
```
id                           0
diagnosis                    0
radius_mean                  0
texture_mean                 0
perimeter_mean               0
area_mean                    0
smoothness_mean              0
compactness_mean             0
concavity_mean               0
concave points_mean          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
```



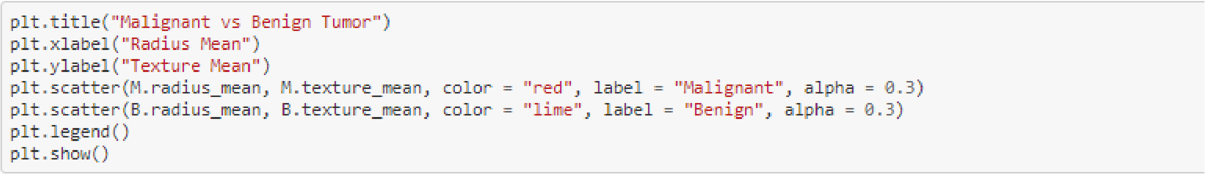
```
## Cleaning the Dataset
df.drop(['Unnamed: 32'], axis=1, inplace = True)
df.drop(['id'], axis=1, inplace = True)
df.columns
```

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

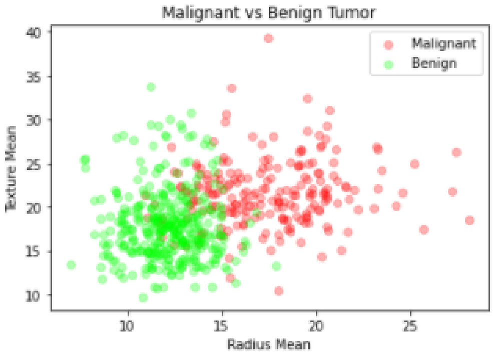
VISUALIZATION:



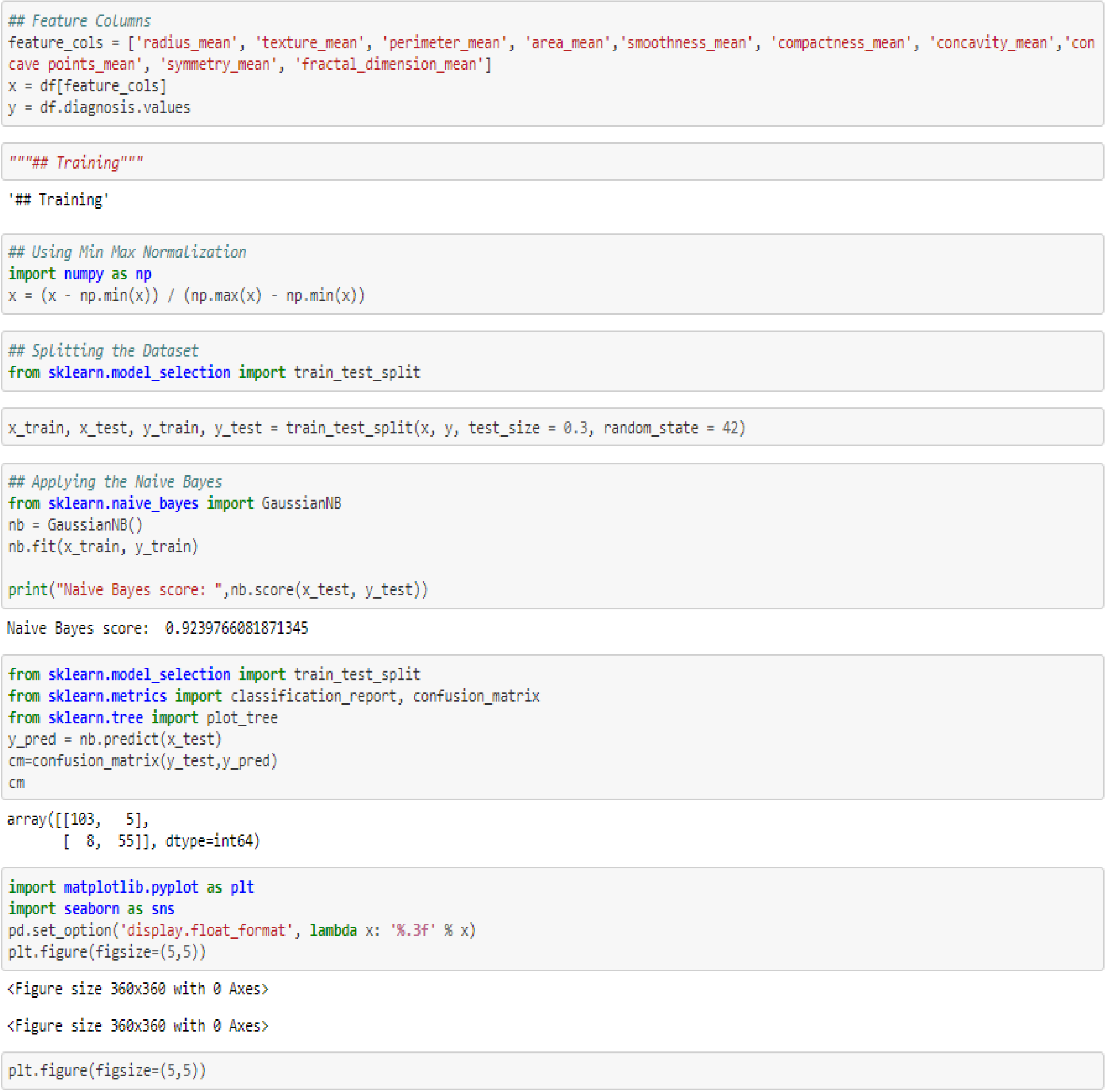
```
## Visualizing
M = df[df.diagnosis == "M"]
B = df[df.diagnosis == "B"]
```



```
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:



```
## Feature Columns
feature_cols = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean','smoothness_mean', 'compactness_mean', 'concavity_mean','con
cave points_mean', 'symmetry_mean', 'fractal_dimension_mean']
x = df[feature_cols]
y = df.diagnosis.values
```

```
"""## Training"""
```

```
'## Training'
```

```
## Using Min Max Normalization
import numpy as np
x = (x - np.min(x)) / (np.max(x) - np.min(x))
```

```
## Splitting the Dataset
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 42)
```

```
## 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
```

```
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
```

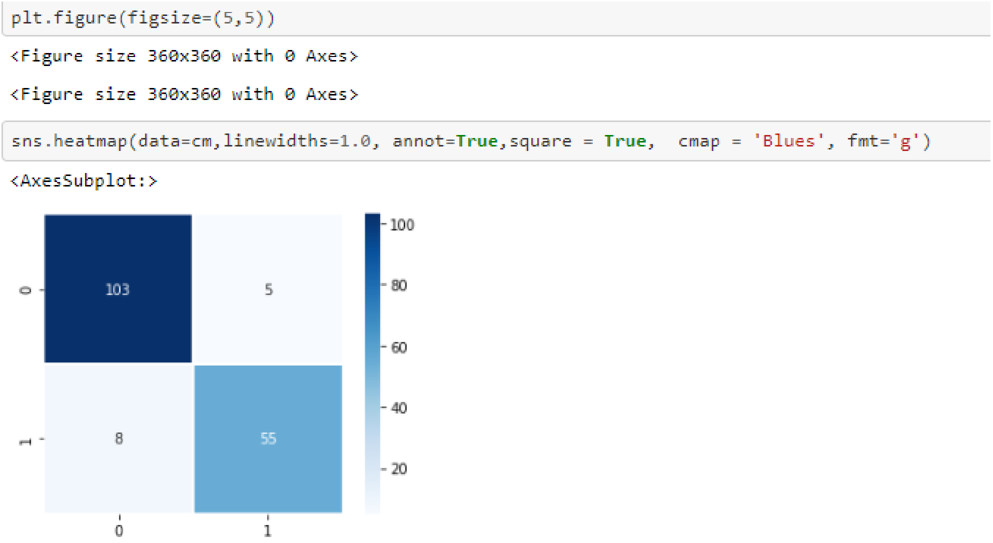
```
array([[103,   5],
       [  8,  55]], dtype=int64)
```

```
import matplotlib.pyplot as plt
import seaborn as sns
pd.set_option('display.float_format', lambda x: '%.3f' % x)
plt.figure(figsize=(5,5))
```

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

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

```
plt.figure(figsize=(5,5))
```



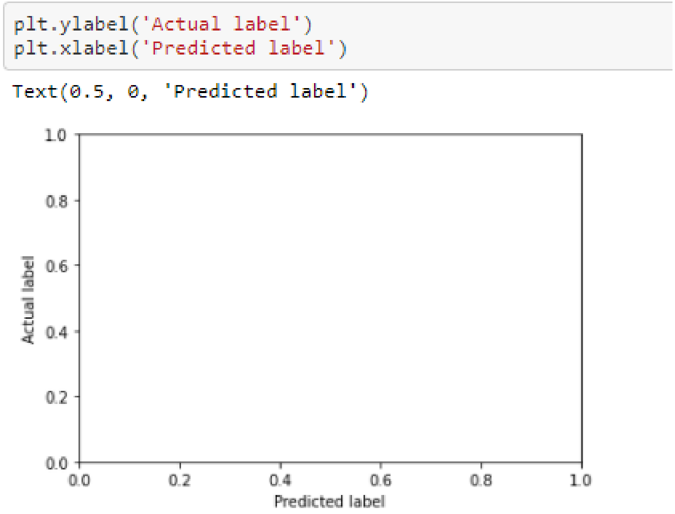
```
plt.figure(figsize=(5,5))
```

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

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

```
sns.heatmap(data=cm,linewidths=1.0, annot=True,square = True,  cmap = 'Blues', fmt='g')
```

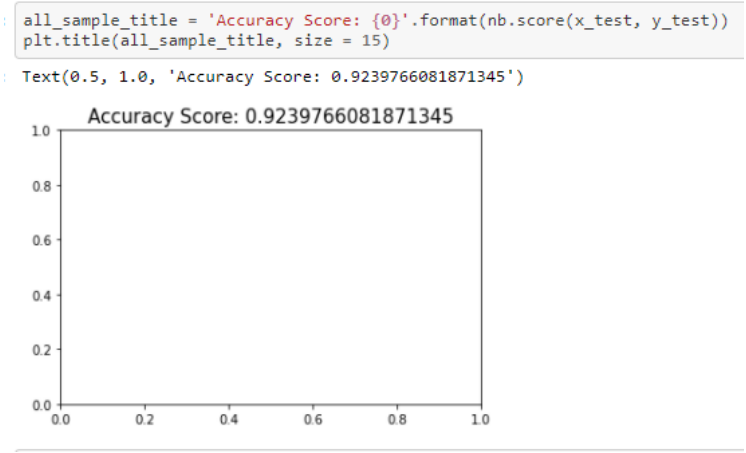
```
<AxesSubplot:>
```



```
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 0, 'Predicted label')
```

FINAL RESULT:



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

```
Text(0.5, 1.0, 'Accuracy Score: 0.9239766081871345')
```

GITHUB LINK:

https://github.com/avnish9898/Ml-Experiment/blob/main/exp-1.ipynb