Avnish Singh jaswal
00113207218
CSE- 1

MACHINE LEARNING

LAB PROGRAM 2

EXPERIMENT-2

Problem Statement

Estimate the accuracy of decision classifier on breast cancer dataset

Algorithm

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute**.**

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3.Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Program Code Snippet

Loading Dataset

```
import pandas as pd

df = pd.read_csv("C:/Users/avnish/Desktop/ml assignment practical files/csv/cancer.csv")
df
```

	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

```
df.head(10)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	te
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	...	
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.08089	...	
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.07400	...	
7	84458202	M	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.05985	...	
8	844981	M	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.09353	...	
9	84501001	M	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.08543	...	

10 rows × 33 columns

```
df.tail()
```

	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

Info About the Data

```
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
```

```
df.shape
```

```
(569, 33)
```

```
df.columns.values
```

```
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)
```

```
df.corr()
```

Preprocessing/Cleaning of dataset.

```
df.corr()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	con points_n
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
area_worst	0.107187	0.941082	0.343546	0.941550	0.959213	0.206718	0.509604	0.675987	0.80
smoothness_worst	0.010338	0.119616	0.077503	0.150549	0.123523	0.805324	0.565541	0.448822	0.45
compactness_worst	-0.002968	0.413463	0.277830	0.455774	0.390410	0.472468	0.865809	0.754968	0.66
concavity_worst	0.023203	0.526911	0.301025	0.563879	0.512606	0.434926	0.816275	0.884103	0.75
concave points_worst	0.035174	0.744214	0.295316	0.771241	0.722017	0.503053	0.815573	0.861323	0.91
symmetry_worst	-0.044224	0.163953	0.105008	0.189115	0.143570	0.394309	0.510223	0.409464	0.37
fractal_dimension_worst	-0.029866	0.007066	0.119205	0.051019	0.003738	0.499316	0.687382	0.514930	0.36
Unnamed: 32	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

```
In [8]: df.isnull().sum()

Out[8]: 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
```

Data Cleaning

```
[60]: 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
12.34    4
12.77    3
15.46    3
12.89    3
13.05    3
        ..
12.31    1
18.81    1
13.30    1
23.09    1
18.25    1
Name: radius_mean, Length: 456, dtype: int64
------------------------*********----------------------------
texture_mean
14.93    3
15.70    3
18.90    3
16.84    3
17.46    3
        ..
20.53    1
17.66    1
24.80    1
20.56    1
10.94    1
Name: texture_mean, Length: 479, dtype: int64
------------------------*********----------------------------
perimeter_mean
82.61     3
134.70    3
87.76     3
130.00    2
58.79     2
         ..
70.21     1
68.69     1
95.55     1
102.90    1
88.52     1
Name: perimeter_mean, Length: 522, dtype: int64
------------------------*********----------------------------
area_mean
512.2     3
1214.0    2
399.8     2
758.6     2
1075.0    2
         ..
704.4     1
904.6     1
646.1     1
300.2     1
1001.0    1
Name: area_mean, Length: 539, dtype: int64
------------------------*********----------------------------
smoothness_mean
0.10070    5
0.10750    4
0.10540    4
0.11500    4
0.10890    3
          ..
0.07274    1
0.07948    1
0.07840    1
0.09780    1
0.07557    1
Name: smoothness_mean, Length: 474, dtype: int64
------------------------*********----------------------------
compactness_mean
0.12060    3
0.11470    3
0.04994    2
0.13390    2
```

```
          ..
0.06698    1
0.11430    1
0.06095    1
0.31140    1
0.18750    1
Name: compactness_mean, Length: 537, dtype: int64
------------------------*********----------------------------
concavity_mean
0.00000    13
0.12040     3
0.01342     2
0.03344     2
0.02688     2
           ..
0.12010     1
0.13480     1
0.05940     1
0.01797     1
0.06593     1
Name: concavity_mean, Length: 537, dtype: int64
------------------------*********----------------------------
concave points_mean
0.000000    13
0.028640     3
0.124200     2
0.052520     2
0.057780     2
            ..
0.053810     1
0.006423     1
0.056020     1
0.066150     1
0.029780     1
Name: concave points_mean, Length: 542, dtype: int64
-------------------------*********----------------------------
symmetry_mean
0.1714    4
0.1769    4
0.1893    4
0.1717    4
0.1601    4
         ..
0.2079    1
0.1671    1
0.2127    1
0.1633    1
0.1382    1
Name: symmetry_mean, Length: 432, dtype: int64
------------------------*********----------------------------
fractal dimension mean
```

```
0.1671    1
0.2127    1
0.1633    1
0.1382    1
Name: symmetry_mean, Length: 432, dtype: int64
------------------------*********----------------------------
fractal_dimension_mean
0.05667    3
0.06113    3
0.05913    3
0.06782    3
0.05907    3
          ..
0.08980    1
0.07421    1
0.06615    1
0.06028    1
0.05044    1
Name: fractal_dimension_mean, Length: 499, dtype: int64
------------------------*********----------------------------
radius_se
0.2860    3
0.2204    3
0.2315    2
0.3380    2
0.3276    2
         ..
0.3428    1
0.5366    1
0.4250    1
0.7661    1
0.2500    1
Name: radius_se, Length: 540, dtype: int64
------------------------*********----------------------------
texture_se
1.3500    3
1.2680    3
0.8561    3
1.1500    3
1.4280    2
         ..
0.8339    1
0.8652    1
1.2000    1
1.9250    1
1.3750    1
Name: texture_se, Length: 519, dtype: int64
------------------------*********----------------------------
perimeter_se
1.778    4
3.564    2
```

```
15.46    1
36.92    1
26.50    1
Name: texture_worst, Length: 511, dtype: int64
-----------------------*********---------------------------
perimeter_worst
101.70    3
105.90    3
117.70    3
104.50    2
152.40    2
          ..
98.87     1
68.62     1
92.12     1
102.20    1
152.50    1
Name: perimeter_worst, Length: 514, dtype: int64
-----------------------*********---------------------------
area_worst
733.5     2
808.9     2
1261.0    2
547.4     2
402.8     2
          ..
741.6     1
750.1     1
392.2     1
697.7     1
1956.0    1
Name: area_worst, Length: 544, dtype: int64
-----------------------*********---------------------------
smoothness_worst
0.12160    4
0.13120    4
0.12750    4
0.12560    4
0.14150    4
           ..
0.18780    1
0.15740    1
0.15510    1
0.13230    1
0.08864    1
Name: smoothness_worst, Length: 411, dtype: int64
-----------------------*********---------------------------
compactness_worst
0.3416    3
0.1486    3
0.1822    2
0.3089    2
```

```
0.08864    1
Name: smoothness_worst, Length: 411, dtype: int64
-----------------------*********---------------------------
compactness_worst
0.3416    3
0.1486    3
0.1822    2
0.3089    2
0.1517    2
          ..
0.7725    1
0.1633    1
0.1885    1
0.1652    1
0.2964    1
Name: compactness_worst, Length: 529, dtype: int64
-----------------------*********---------------------------
concavity_worst
0.00000    13
0.13770     3
0.45040     3
0.18040     2
0.38530     2
           ..
0.18560     1
0.26710     1
0.13900     1
0.18980     1
0.09203     1
Name: concavity_worst, Length: 539, dtype: int64
-----------------------*********---------------------------
concave points_worst
0.00000    13
0.04306     3
0.07431     3
0.05556     3
0.12180     3
           ..
0.09744     1
0.08436     1
0.19390     1
0.09331     1
0.04970     1
Name: concave points_worst, Length: 492, dtype: int64
-----------------------*********---------------------------
symmetry_worst
0.2226    3
0.2369    3
0.2383    3
0.2972    3
0.3196    3
```

```
          ..
0.09744     1
0.08436     1
0.19390     1
0.09331     1
0.04970     1
Name: concave points_worst, Length: 492, dtype: int64
-----------------------*********----------------------------
symmetry_worst
0.2226    3
0.2369    3
0.2383    3
0.2972    3
0.3196    3
         ..
0.3013    1
0.2356    1
0.3322    1
0.3591    1
0.2500    1
Name: symmetry_worst, Length: 500, dtype: int64
-----------------------*********----------------------------
fractal_dimension_worst
0.07427    3
0.12970    2
0.07918    2
0.08633    2
0.09136    2
          ..
0.07948    1
0.06818    1
0.07320    1
0.07247    1
0.09981    1
Name: fractal_dimension_worst, Length: 535, dtype: int64
-----------------------*********----------------------------
Unnamed: 32
Series([], Name: Unnamed: 32, dtype: int64)
-----------------------*********----------------------------
```

```
df['diagnosis'].value_counts()
```

```
B    357
M    212
Name: diagnosis, dtype: int64
```

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

	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 × 32 columns

Visualization

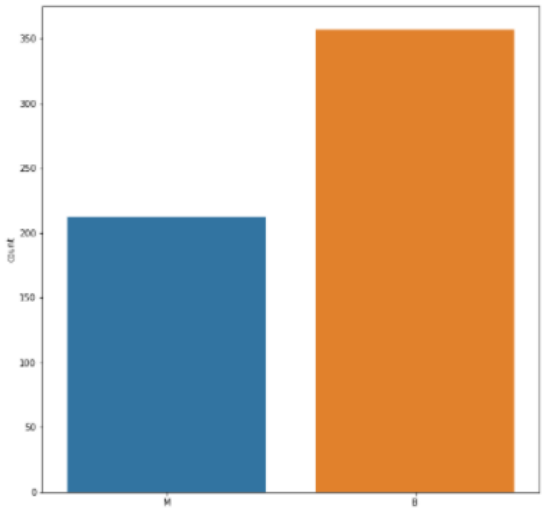
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
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
```

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

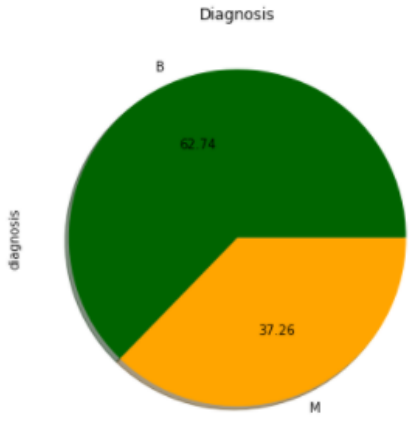
```
C:\Users\avnish\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword
arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit
eyword will result in an error or misinterpretation.
  warnings.warn(
```




```
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
```

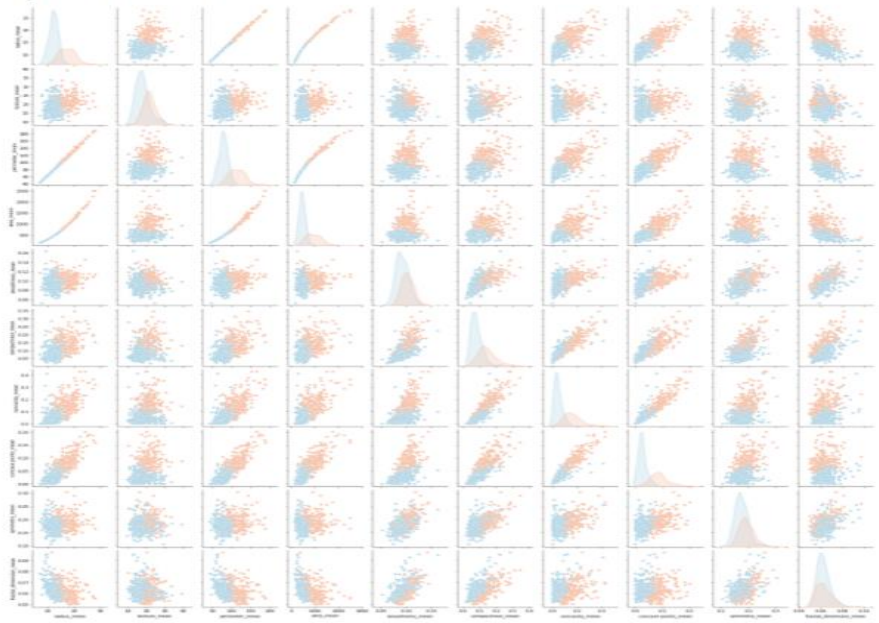
```
df.diagnosis.value_counts().plot(kind='pie',shadow=True,colors=('darkgreen','orange'),autopct='%.2f',figsize=(8,6))
plt.title('Diagnosis')
plt.show()
```



```
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')
```

```
<seaborn.axisgrid.PairGrid at 0x1751e3c4a30>
```

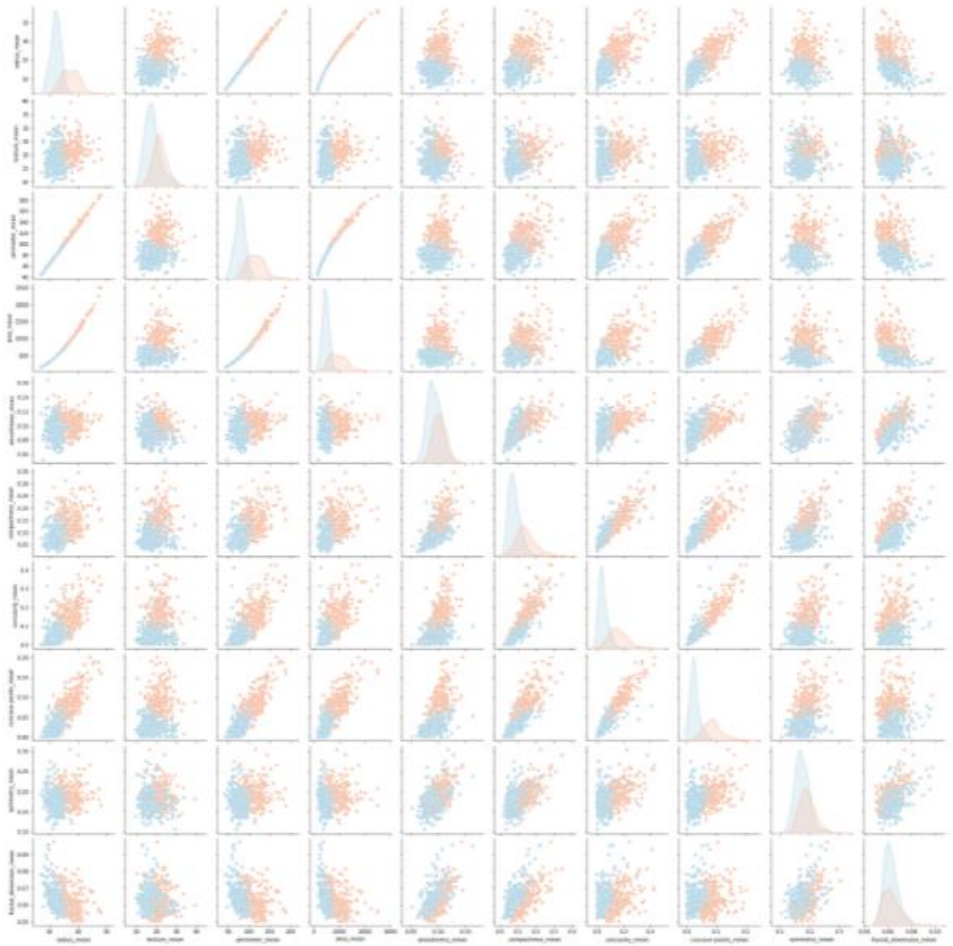
```
<Figure size 720x720 with 0 Axes>
```




```
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')
```


```
<seaborn.axisgrid.PairGrid at 0x17522046fd0>
```

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

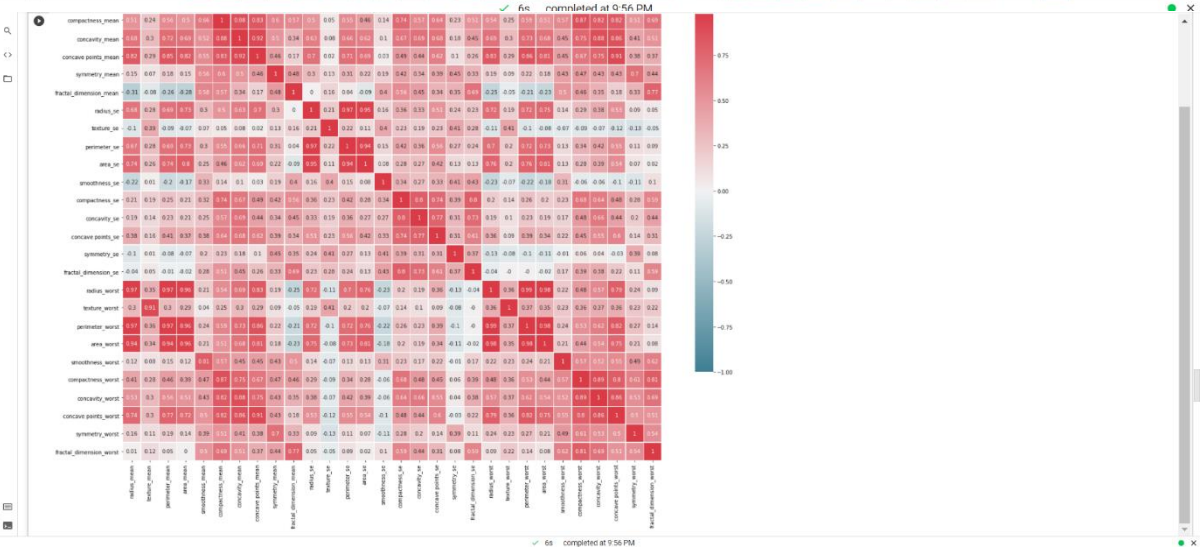


```
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True)

plt.tight_layout()
```

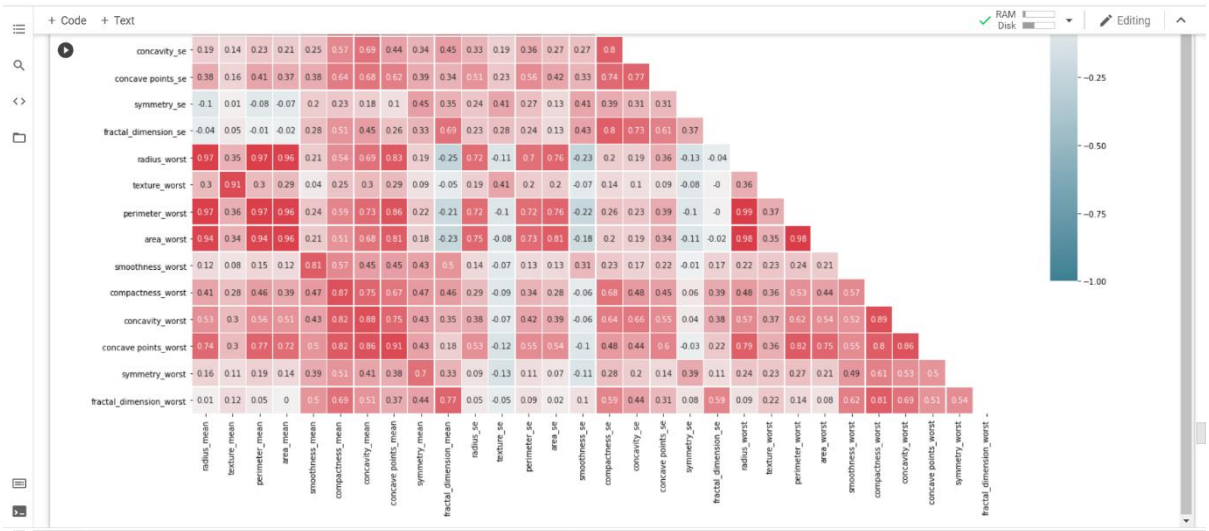
✓ 6s completed at 9:56 PM

```
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()
```

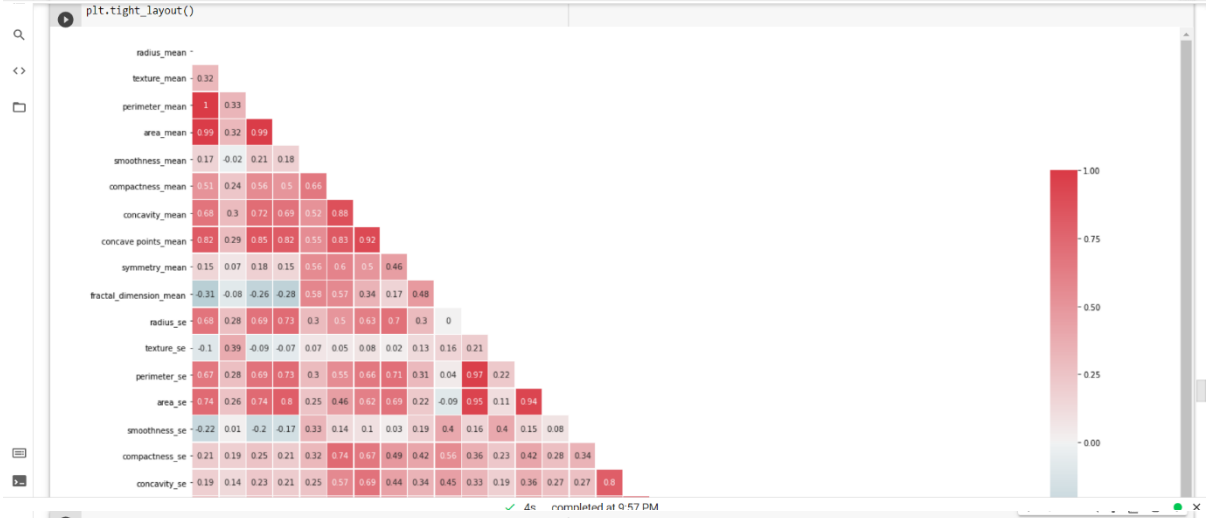
✓ 6s completed at 9:56 PM



✓ 6s completed at 9:56 PM

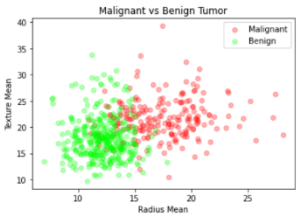



```
plt.tight_layout()
```



4s completed at 9:57 PM

```
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()
```



Algorithm Implementation

```
[28] feature_cols = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean','smoothness_mean', 'compactness_mean', 'concavity_mean','concave points_mean', 'symmetry_mean', 'fractal_c
```

```
[29] x = df[feature_cols]
     y = df.diagnosis.values
```

```
[30] x.head()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883

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

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
0	0.521037	0.022658	0.545989	0.363733	0.593753	0.792037	0.703140	0.731113	0.686364	0.605518
1	0.643144	0.272574	0.615783	0.501591	0.289880	0.181768	0.203608	0.348757	0.379798	0.141323
2	0.601496	0.390260	0.595743	0.449417	0.514309	0.431017	0.462512	0.635686	0.509596	0.211247
3	0.210090	0.360839	0.233501	0.102906	0.811321	0.811361	0.565604	0.522863	0.776263	1.000000
4	0.629893	0.156578	0.630986	0.489290	0.430351	0.347893	0.463918	0.518390	0.378283	0.186816
...	...	...	...	...	...	...	...	...	...	...
564	0.690000	0.428813	0.678668	0.566490	0.526948	0.296055	0.571462	0.690358	0.336364	0.132056
565	0.622320	0.626987	0.604036	0.474019	0.407782	0.257714	0.337395	0.486630	0.349495	0.113100
566	0.455251	0.621238	0.445788	0.303118	0.288165	0.254340	0.216753	0.263519	0.267677	0.137321
567	0.644564	0.663510	0.665538	0.475716	0.588336	0.790197	0.823336	0.755467	0.675253	0.425442
568	0.036869	0.501522	0.028540	0.015907	0.000000	0.074351	0.000000	0.000000	0.266162	0.187026

569 rows × 10 columns

```
from sklearn.model_selection import train_test_split

#for checking testing results
```


[31]

4	0.629893	0.156578	0.630986	0.489290	0.430351	0.347893	0.463918	0.518390	0.378283	0.186816
...	...	...	...	...	...	...	...	...	...	...
564	0.690000	0.428813	0.678668	0.566490	0.526948	0.296055	0.571462	0.690358	0.336364	0.132056
565	0.622320	0.626987	0.604036	0.474019	0.407782	0.257714	0.337395	0.486630	0.349495	0.113100
566	0.455251	0.621238	0.445788	0.303118	0.288165	0.254340	0.216753	0.263519	0.267677	0.137321
567	0.644564	0.663510	0.665538	0.475716	0.588336	0.790197	0.823336	0.755467	0.675253	0.425442
568	0.036869	0.501522	0.028540	0.015907	0.000000	0.074351	0.000000	0.000000	0.266162	0.187026

569 rows × 10 columns


```
from sklearn.model_selection import train_test_split

#for checking testing results
from sklearn.metrics import classification_report, confusion_matrix

#for visualizing tree
from sklearn.tree import plot_tree

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)

print("Training split input- ", x_train.shape)
print("Testing split input- ", x_test.shape)
```

```
Training split input-  (455, 10)
Testing split input-  (114, 10)
```

```
[33] from sklearn.tree import DecisionTreeClassifier
```


```
[32] #for checking testing results
     from sklearn.metrics import classification_report, confusion_matrix

     #for visualizing tree
     from sklearn.tree import plot_tree

     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)

     print("Training split input- ", x_train.shape)
     print("Testing split input- ", x_test.shape)
```

```
Training split input-  (455, 10)
Testing split input-  (114, 10)
```

```
[33] from sklearn.tree import DecisionTreeClassifier
```


```
[34] dt = DecisionTreeClassifier()
```

```
dt.fit(x_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Final Graph/ROC/Confusion Matrix

```
[36] y_pred = dt.predict(x_test)
     print("Classification report - \n", classification_report(y_test,y_pred))
```

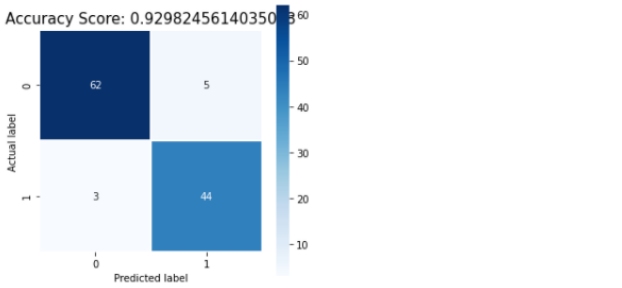
```
Classification report -
              precision    recall  f1-score   support

           B       0.94      0.93      0.93        67
           M       0.90      0.91      0.91        47

    accuracy                           0.92       114
   macro avg       0.92      0.92      0.92       114
weighted avg       0.92      0.92      0.92       114
```

```
[37] cm=confusion_matrix(y_test,y_pred)
     cm
```

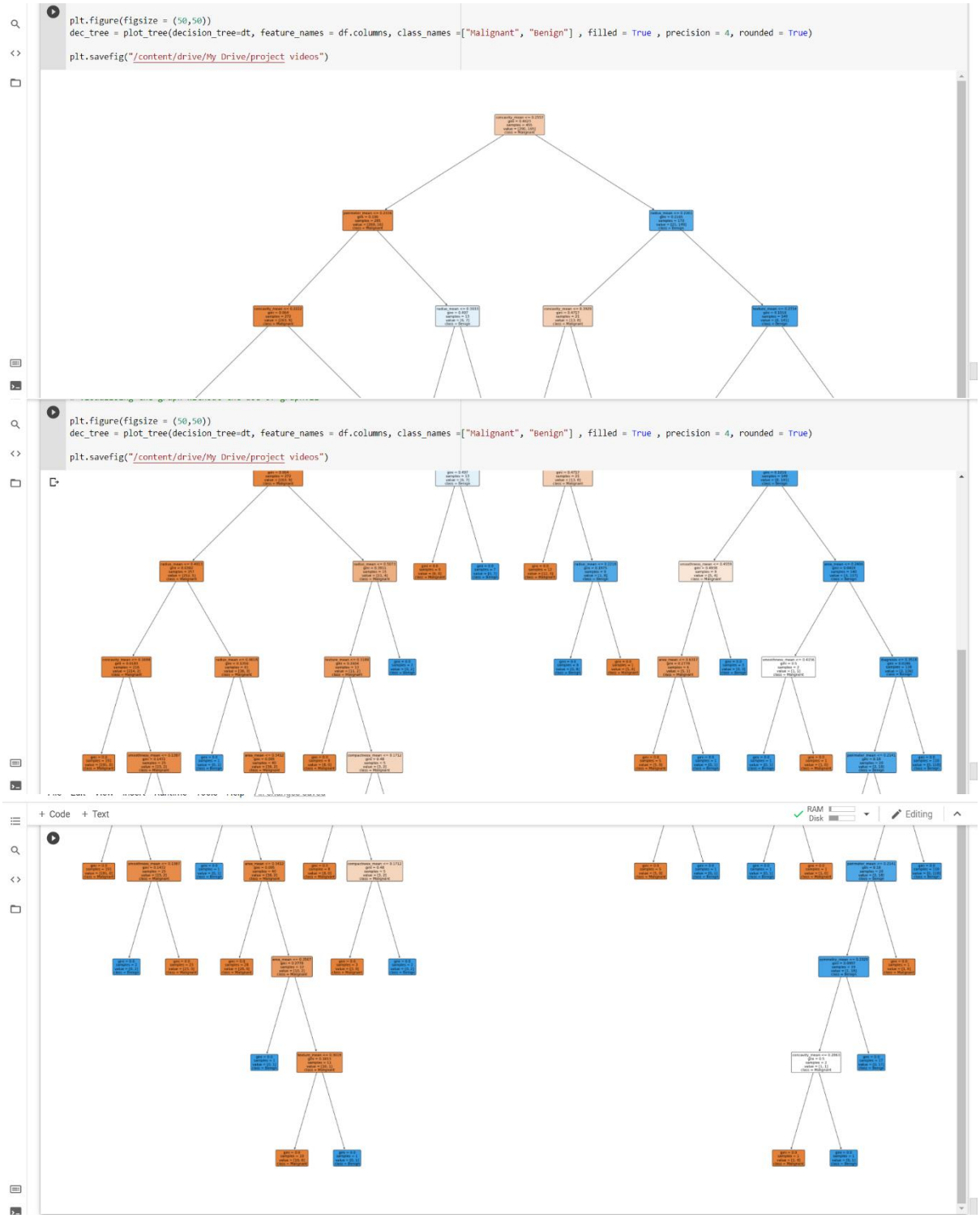
```
array([[62,  5],
       [ 4, 43]])
```

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

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

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

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

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

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

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

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

plt.savefig("C:/Users/avnish/Desktop/ml assignment practical files/accu.png")
```




```
pip install graphviz
```



Github Link

https://github.com/avnish9898/Ml-Experiment/blob/main/exp2.ipynb