

Name- Akjot Singh

Enroll- 00213202717

Class- CSE-1

EXPERIMENT-3

Problem Statement

Estimate the accuracy of decision classifier on breast cancer dataset using 5 fold cross validation

Algorithm

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree.

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

This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

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

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further.

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

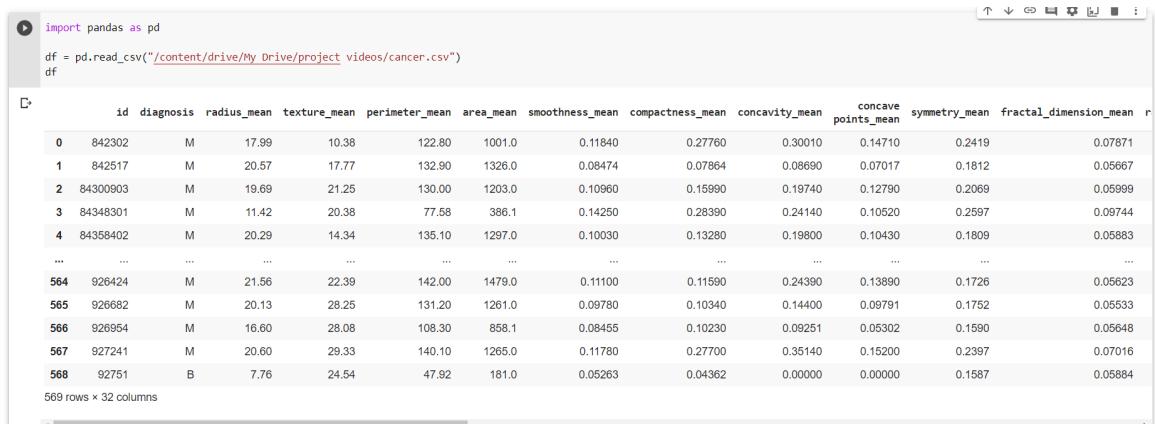
Step-4: Generate the decision tree node, which contains the best attribute.

It continues the process until it reaches the leaf node of the tree.

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 it 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("/content/drive/My_Drive/project_videos/cancer.csv")
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean	symmetry_mean	fractal_dimension_mean	r
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	
...	
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	

```

          id diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean concave
      0 842302      M    17.99     10.38     122.80   1001.0    0.11840    0.27760    0.30010    0.14710    0.2419    0.07871
      1 842517      M    20.57     17.77     132.90   1326.0    0.08474    0.07864    0.08690    0.07017    0.1812    0.05667
      2 84300903     M    19.69     21.25     130.00   1203.0    0.10860    0.15990    0.19740    0.12790    0.2069    0.05999
      3 84348301     M    11.42     20.38      77.58    386.1    0.14250    0.28390    0.24140    0.10520    0.2597    0.09744
      4 84358402     M    20.29     14.34     135.10   1297.0    0.10030    0.13280    0.19800    0.10430    0.1809    0.05883
      5 843786      M    12.45     15.70      82.57    477.1    0.12780    0.17000    0.15780    0.08089    0.2087    0.07613
      6 844359      M    18.25     19.98     119.60   1040.0    0.09463    0.10900    0.11270    0.07400    0.1794    0.05742
      7 84458202     M    13.71     20.83      90.20    577.9    0.11890    0.16450    0.09366    0.05985    0.2196    0.07451
      8 844981      M    13.00     21.82      87.50    519.8    0.12730    0.19320    0.18590    0.09353    0.2350    0.07389
      9 84501001     M    12.46     24.04      83.97    475.9    0.11860    0.23960    0.22730    0.08543    0.2030    0.08243

#to read the last end of data
df.tail()

          id diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean concave
  564 926424      M    21.56     22.39     142.00   1479.0    0.11100    0.11590    0.24390    0.13890    0.1726    0.05623
  565 926682      M    20.13     28.25     131.20   1261.0    0.09780    0.10340    0.14400    0.09791    0.1752    0.05533
  566 926954      M    16.60     28.08     108.30   858.1    0.08455    0.10230    0.09251    0.05302    0.1590    0.05648

[4] 4 84358402     M    20.29     14.34     135.10   1297.0    0.10030    0.13280    0.19800    0.10430    0.1809    0.05883
  5 843786      M    12.45     15.70      82.57    477.1    0.12780    0.17000    0.15780    0.08089    0.2087    0.07613
  6 844359      M    18.25     19.98     119.60   1040.0    0.09463    0.10900    0.11270    0.07400    0.1794    0.05742
  7 84458202     M    13.71     20.83      90.20    577.9    0.11890    0.16450    0.09366    0.05985    0.2196    0.07451
  8 844981      M    13.00     21.82      87.50    519.8    0.12730    0.19320    0.18590    0.09353    0.2350    0.07389
  9 84501001     M    12.46     24.04      83.97    475.9    0.11860    0.23960    0.22730    0.08543    0.2030    0.08243

#to read the last end of data
df.tail()

          id diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean concave
  564 926424      M    21.56     22.39     142.00   1479.0    0.11100    0.11590    0.24390    0.13890    0.1726    0.05623
  565 926682      M    20.13     28.25     131.20   1261.0    0.09780    0.10340    0.14400    0.09791    0.1752    0.05533
  566 926954      M    16.60     28.08     108.30   858.1    0.08455    0.10230    0.09251    0.05302    0.1590    0.05648
  567 927241      M    20.60     29.33     140.10   1265.0    0.11780    0.27700    0.35140    0.15200    0.2397    0.07016
  568 92751       B    7.76     24.54      47.92    181.0    0.05263    0.04362    0.00000    0.00000    0.1587    0.05884

df.info()

<class 'pandas.core.frame.DataFrame'
RangeIndex: 568 entries, 0 to 568
Data columns (total 32 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
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB

[8] df.shape
(569, 32)

#print all the columns of dataset
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'], dtype=object)

```

Preprocessing/Cleaning of dataset

[10] df.corr()

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean	symmetry_mean	fractal_dimension_mean
id	1.000000	0.074626	0.099770	0.073159	0.096893	-0.012968	0.000096	0.050080	0.044158	-0.022114	-0.01
radius_mean	0.074626	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822529	0.147741	-0.3
texture_mean	0.099770	0.323782	1.000000	0.329533	0.321066	-0.023389	0.236702	0.302418	0.293464	0.071401	-0.01
perimeter_mean	0.073159	0.997855	0.329533	1.000000	0.986507	0.202728	0.556936	0.716136	0.850977	0.183027	-0.2
area_mean	0.096893	0.987357	0.321066	0.986507	1.000000	0.177028	0.498502	0.685983	0.823269	0.151293	-0.2
smoothness_mean	-0.012968	0.170581	-0.023389	0.202728	0.177028	1.000000	0.659123	0.521984	0.553695	0.557775	0.5
compactness_mean	0.000096	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831135	0.602641	0.5
concavity_mean	0.050080	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921391	0.500667	0.3
concave_points_mean	0.044158	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000000	0.462497	0.1
symmetry_mean	-0.022114	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	1.000000	0.4
fractal_dimension_mean	-0.052511	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166917	0.479921	1.0
radius_se	0.143048	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.698050	0.303379	0.01
texture_se	-0.007526	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.021480	0.128053	0.1
perimeter_se	0.137331	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.710650	0.313893	0.0
area_se	0.177742	0.735864	0.259645	0.744983	0.800086	0.246552	0.455653	0.617427	0.690299	0.223970	-0.0
smoothness_se	0.096781	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564	0.027653	0.187321	0.4
compactness_se	0.033961	0.206000	0.191975	0.250740	0.212583	0.318943	0.738722	0.670279	0.490424	0.421659	0.5
concavity_se	0.055239	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270	0.439167	0.342627	0.4

✓ 0s completed at 9:51 PM

• Finding Relationships:

A great aspect of the Pandas module is the corr() method.

The corr() method calculates the relationship between each column in your data set.

df.corr()

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean	symmetry_mean	fractal_dimension_mean	radius_worst	texture_worst	perimeter_worst	area_worst
id	1.000000	0.074626	0.099770	0.073159	0.096893	-0.012968	0.000096	0.050080	0.044158	-0.022114	-0.01	0.257265	0.082465	0.084704	0.070680
radius_mean	0.074626	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822529	0.147741	-0.3	0.257265	0.082465	0.084704	0.070680
texture_mean	0.099770	0.323782	1.000000	0.329533	0.321066	-0.023389	0.236702	0.302418	0.293464	0.071401	-0.01	0.257265	0.082465	0.084704	0.070680
perimeter_mean	0.073159	0.997855	0.329533	1.000000	0.986507	0.177028	0.498502	0.659123	0.521984	0.147741	-0.01	0.257265	0.082465	0.084704	0.070680
area_mean	0.096893	0.987357	0.321066	0.986507	1.000000	0.177028	0.498502	0.659123	0.521984	0.147741	-0.01	0.257265	0.082465	0.084704	0.070680
smoothness_mean	-0.012968	0.170581	-0.023389	0.202728	0.177028	1.000000	0.659123	0.521984	0.147741	-0.01	-0.01	0.257265	0.082465	0.084704	0.070680
compactness_mean	0.000096	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831135	0.602641	0.5	0.257265	0.082465	0.084704	0.070680
concavity_mean	0.050080	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921391	0.500667	0.3	0.257265	0.082465	0.084704	0.070680
concave_points_mean	0.044158	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000000	0.462497	0.1	0.257265	0.082465	0.084704	0.070680
symmetry_mean	-0.022114	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	1.000000	0.4	0.257265	0.082465	0.084704	0.070680
fractal_dimension_mean	-0.052511	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166917	0.479921	1.000000	0.257265	0.082465	0.084704	0.070680
radius_se	0.143048	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.698050	0.303379	0.01	0.257265	0.082465	0.084704	0.070680
texture_se	-0.007526	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.021480	0.128053	0.1	0.257265	0.082465	0.084704	0.070680
perimeter_se	0.137331	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.710650	0.313893	0.0	0.257265	0.082465	0.084704	0.070680
area_se	0.177742	0.735864	0.259645	0.744983	0.800086	0.246552	0.455653	0.617427	0.147741	-0.01	-0.01	0.257265	0.082465	0.084704	0.070680
smoothness_se	0.096781	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564	0.027653	0.187321	0.4	0.257265	0.082465	0.084704	0.070680
compactness_se	0.033961	0.206000	0.191975	0.250740	0.212583	0.318943	0.738722	0.670279	0.490424	0.421659	0.5	0.257265	0.082465	0.084704	0.070680
concavity_se	0.055239	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270	0.439167	0.342627	0.4	0.257265	0.082465	0.084704	0.070680
symmetry_se	-0.022114	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	0.449137	0.4	0.257265	0.082465	0.084704	0.070680
fractal_dimension_se	-0.052511	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166917	0.479921	0.4	0.257265	0.082465	0.084704	0.070680
radius_worst	0.143048	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.698050	0.303379	0.01	0.257265	0.082465	0.084704	0.070680
texture_worst	-0.007526	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.021480	0.128053	0.1	0.257265	0.082465	0.084704	0.070680
perimeter_worst	0.137331	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.710650	0.313893	0.0	0.257265	0.082465	0.084704	0.070680
area_worst	0.177742	0.735864	0.259645	0.744983	0.800086	0.246552	0.455653	0.617427	0.147741	-0.01	-0.01	0.257265	0.082465	0.084704	0.070680
smoothness_worst	0.096781	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564	0.027653	0.187321	0.4	0.257265	0.082465	0.084704	0.070680
compactness_worst	0.033961	0.206000	0.191975	0.250740	0.212583	0.318943	0.738722	0.670279	0.490424	0.421659	0.5	0.257265	0.082465	0.084704	0.070680
concavity_worst	0.055239	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270	0.439167	0.342627	0.4	0.257265	0.082465	0.084704	0.070680
symmetry_worst	-0.022114	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	0.449137	0.4	0.257265	0.082465	0.084704	0.070680
fractal_dimension_worst	-0.052511	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166917	0.479921	0.4	0.257265	0.082465	0.084704	0.070680
radius_worst	0.143048	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.698050	0.303379	0.01	0.257265	0.082465	0.084704	0.070680
texture_worst	-0.007526	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.021480	0.128053	0.1	0.257265	0.082465	0.084704	0.070680
perimeter_worst	0.137331	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.710650	0.313893	0.0	0.257265	0.082465	0.084704	0.070680
area_worst	0.177742	0.735864	0.259645	0.744983	0.800086	0.246552	0.455653	0.617427	0.147741	-0.01	-0.01	0.257265	0.082465	0.084704	0.070680
smoothness_worst	0.096781	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564	0.027653	0.187321	0.4	0.257265	0.082465	0.084704	0.070680
compactness_worst	0.033961	0.206000	0.191975	0.250740	0.212583	0.318943	0.738722	0.670279	0.490424	0.421659	0.5	0.257265	0.082465	0.084704	0.070680
concavity_worst	0.055239	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270	0.439167	0.342627	0.4	0.257265	0.082465	0.084704	0.070680
symmetry_worst	-0.022114	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	0.449137	0.4	0.257265	0.082465	0.084704	0.070680

```

for i in df.columns:
    print(i)
    print(df[i].value_counts())
    print('-----')
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
1.143     2
1.535     2
2.406     2
...
2.629     1
4.675     1
3.475     1
1.527     1
2.000     1
Name: perimeter_se, Length: 533, dtype: int64
-----
area_se
17.67     3

```

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

```

B    357
M    212
Name: diagnosis, dtype: int64

```

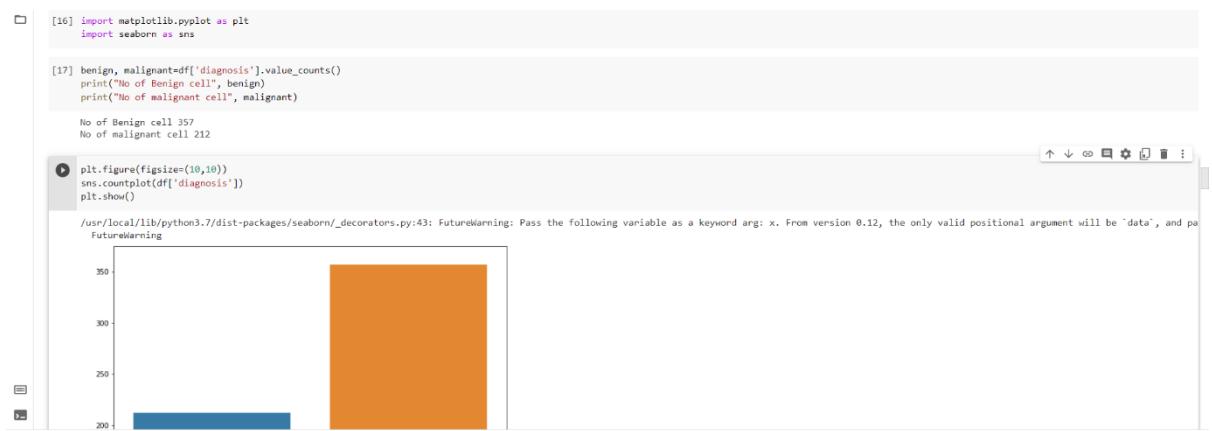
[14] df=df.drop(['id'], axis = 1)
df

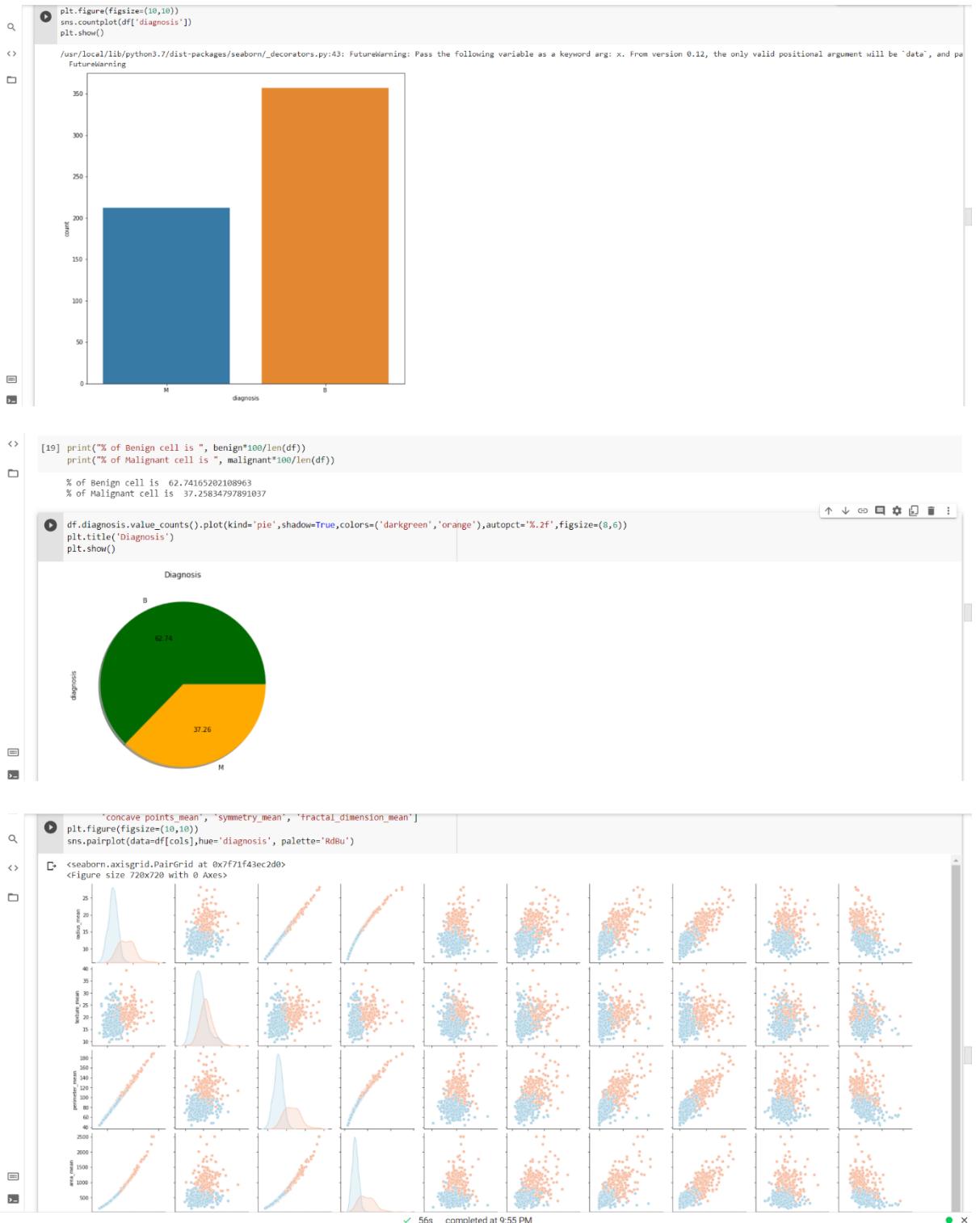
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean	symmetry_mean	fractal_dimension_mean	radius_se
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	1.0950
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	0.5435
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	0.7456
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	0.4956
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	0.7572
...
564	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	1.1760
565	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	0.7655
566	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	0.4564
567	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	0.7260
568	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	0.3857

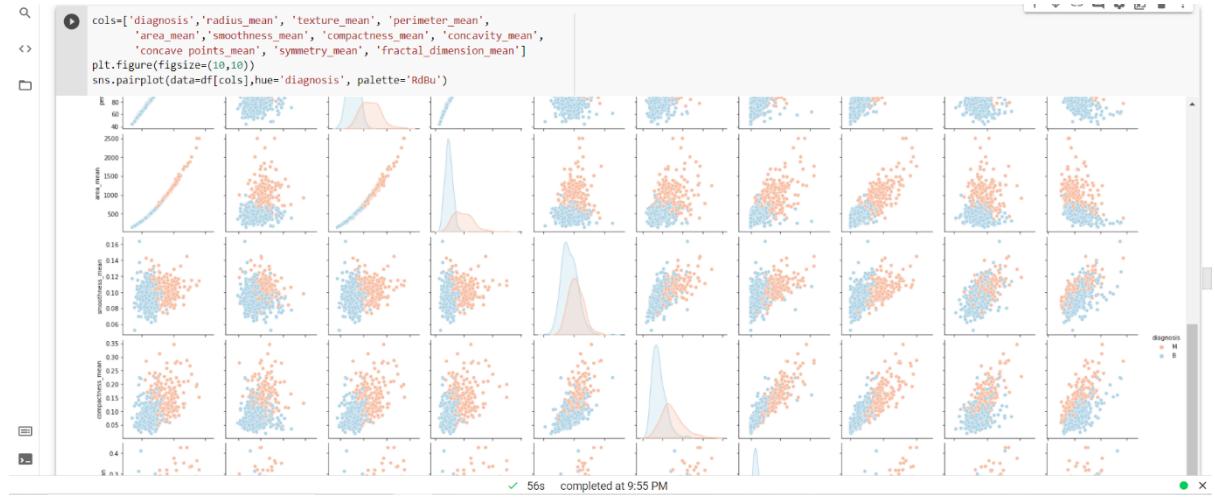
569 rows × 31 columns

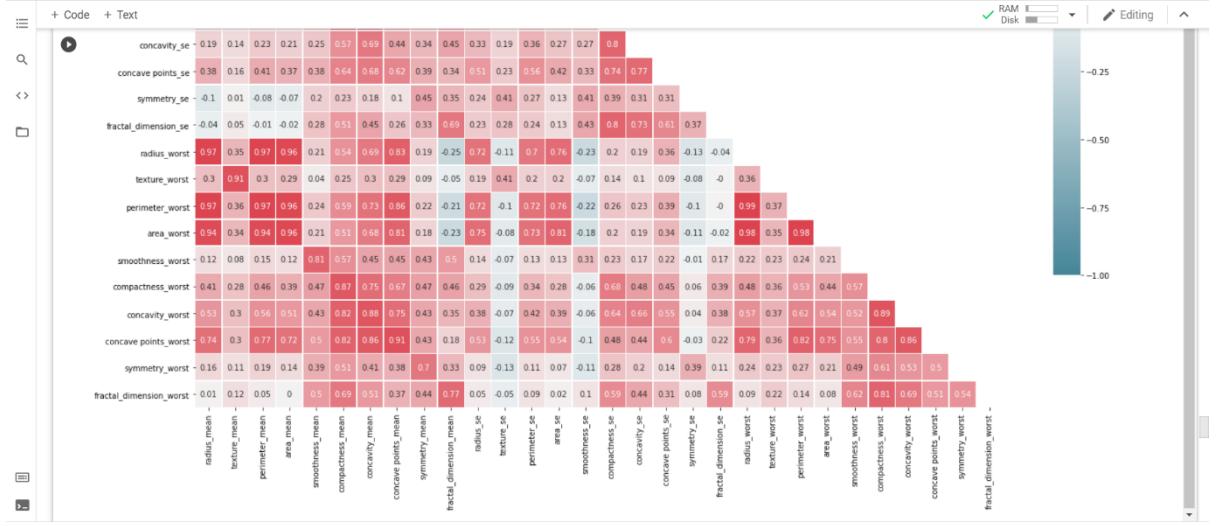
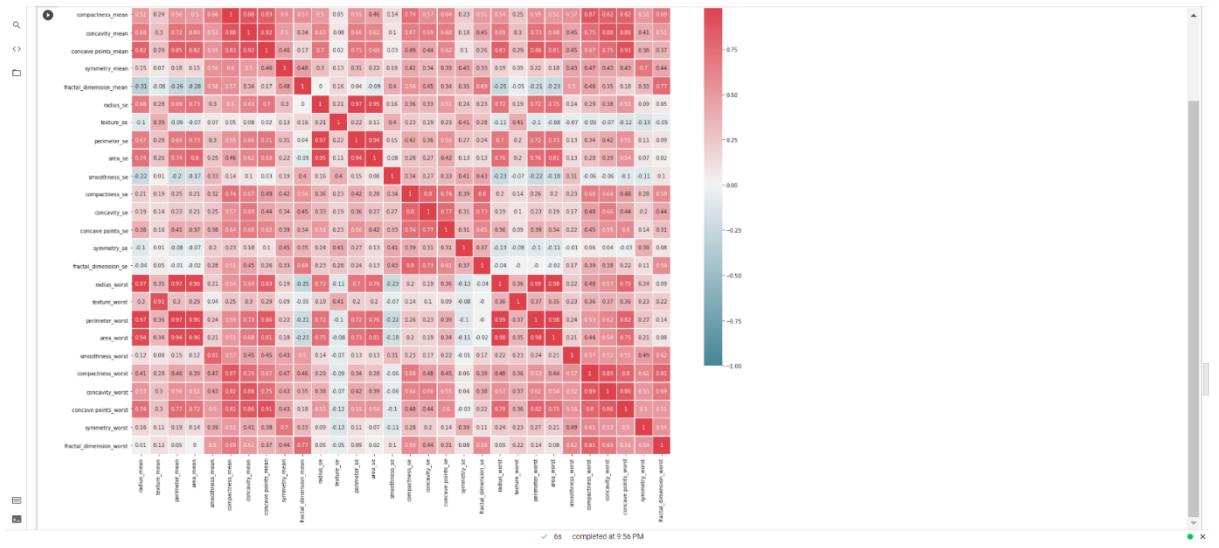
✓ 0s completed at 9:53 PM

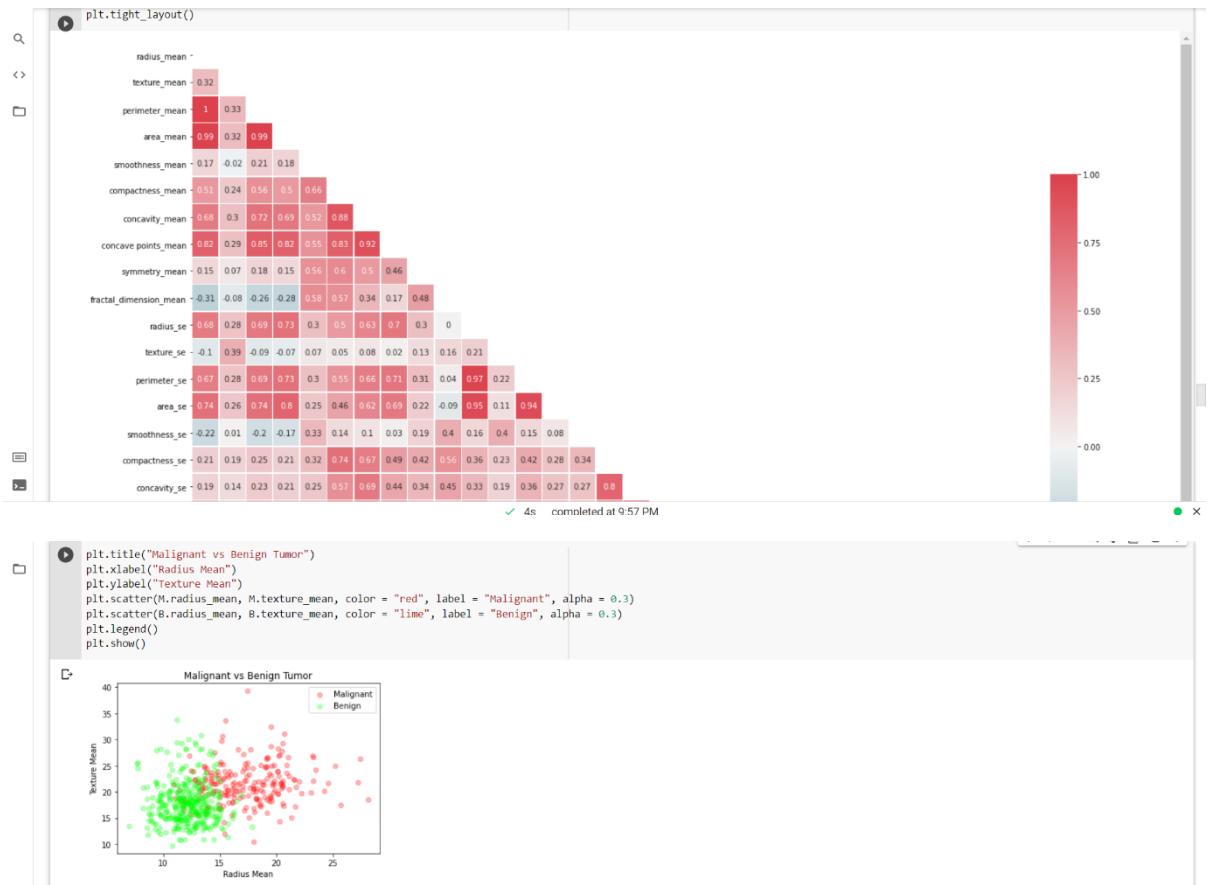
Visualization











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_dimension_mean']
[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          17.98        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	0.811381	0.5665604	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	0.4565251	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 x 10 columns

```
[32] 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
Q	
<>	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	0.486330	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

```

[31]   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)

[32]   from sklearn.tree import DecisionTreeClassifier
      #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
      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

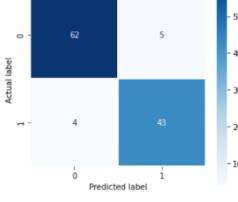
[37] cm=confusion_matrix(y_test,y_pred)
cm
array([[62,  5],
       [ 4, 43]])

❶ 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')

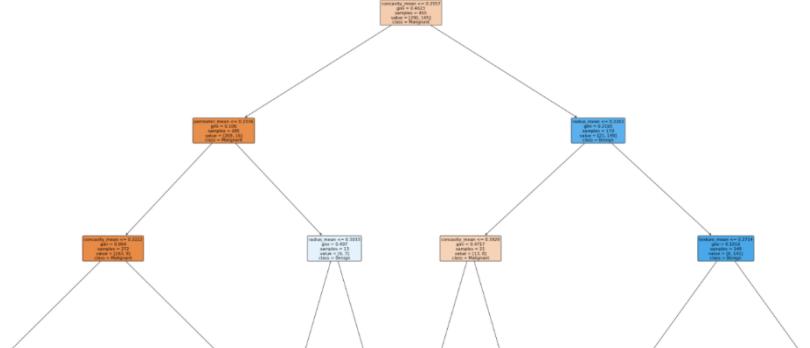
all_sample_title = 'Accuracy Score: {}'.format(dt.score(x_test, y_test))
plt.title(all_sample_title, size = 15)

❷ 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')

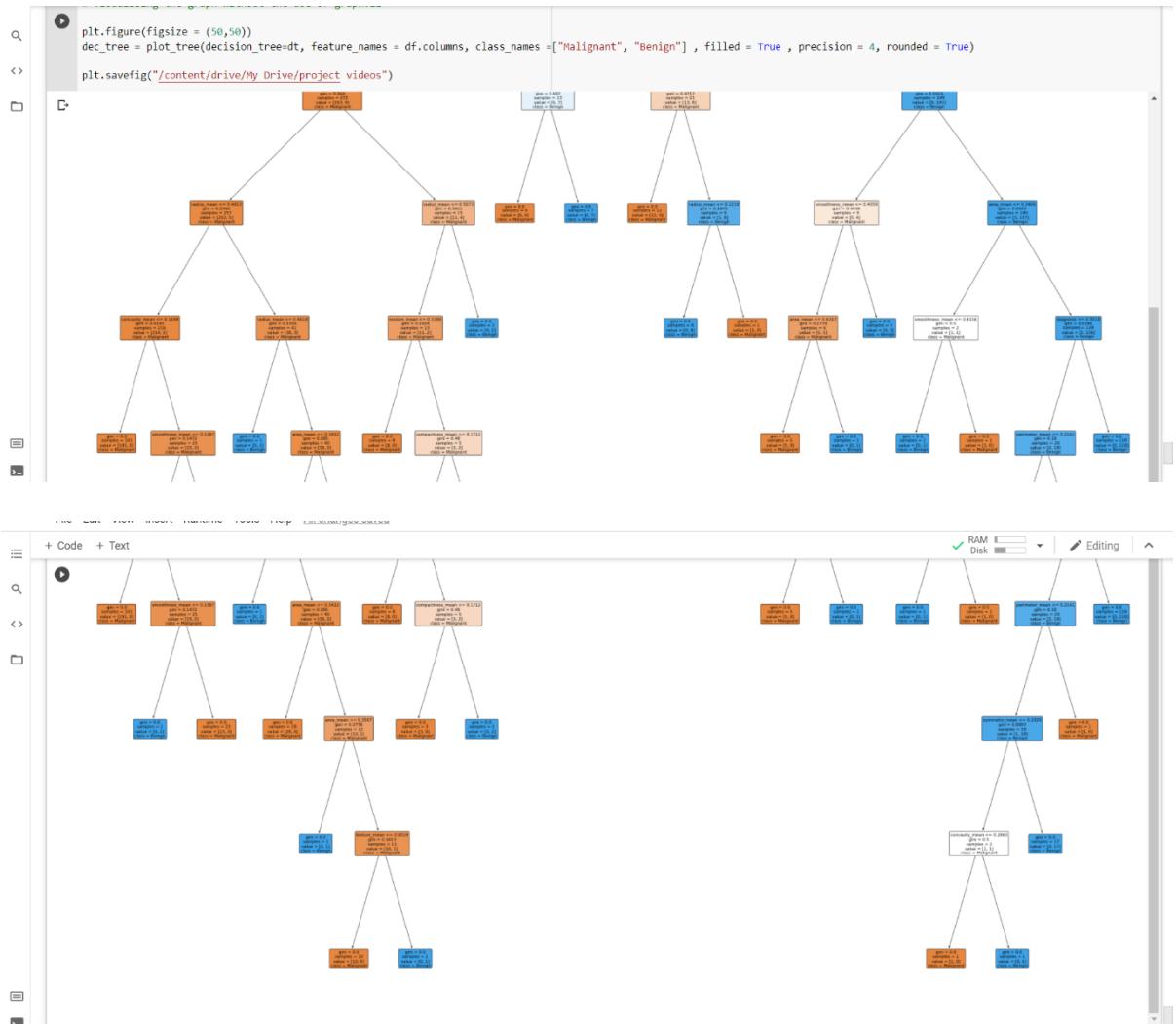
all_sample_title = 'Accuracy Score: {}'.format(dt.score(x_test, y_test))
plt.title(all_sample_title, size = 15)
plt.savefig("/content/drive/My Drive/project videos")

❸ Accuracy Score: 0.92105263157894

plt.savefig("/content/drive/My Drive/project videos")

[40] ! pip install graphviz

❹ plt.figure(figsize = (50,50))
dec_tree = plot_tree(decision_tree=dt, feature_names = df.columns, class_names =["Malignant", "Benign"] , filled = True , precision = 4, rounded = True)
plt.savefig("/content/drive/My Drive/project videos")


```



Accuracy Using K-Fold Method

```
from sklearn.model_selection import cross_val_score,KFold
from numpy import array
kfold = KFold(n_splits=5, random_state=None, shuffle=False)
z = cross_val_score(dt, x, y, cv=cv)
np.mean(z)
```

Github Link

https://github.com/AkSingh03/MACHINE_LEARNING/blob/main/Experimnet_2.ipynb