- Machine Learning Lab3
- Topic: Support Vector Machines
- Student Name: Jyosna Philip
- Roll No: 19
- Date: 14th March '24
- Submission : 20th March '24

In [ ]:

```
import pandas as pd
import numpy as np
from sklearn import svm,tree
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

```
C:\Users\jyosn\AppData\Local\Temp\ipykernel_3364\3362773470.py:1: DeprecationWarn
ing:
Pyarrow will become a required dependency of pandas in the next major release of
pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better i
nteroperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

1. Run SVM2 for cancer.csv

In [ ]:

```
# reading csv file and extracting class column to y.
x = pd.read_csv("cancer.csv")
a = np.array(x)
print(x.head())
y = a[:,2] # classes having 0 and 1  #gets all rows of 3rd column

# extracting two features
x = np.column_stack((x.col1,x.col2))

# 569 samples and 2 features
print(x.shape)

print (x)
print(y)
```

```
     col1    col2  diagnosis  Unnamed: 3
0  122.80  1001.0          1         NaN
1  132.90  1326.0          1         NaN
2  130.00  1203.0          1         NaN
3   77.58   386.1          1         NaN
4  135.10  1297.0          1         NaN
(569, 2)
[[ 122.8  1001.  ]
 [ 132.9  1326.  ]
 [ 130.   1203.  ]
 ...
 [ 108.3   858.1 ]
 [ 140.1  1265.  ]
 [  47.92  181.  ]]
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1.
 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 0. 1. 0.
 1. 1. 0. 1. 0. 1. 1. 0. 0. 0. 1. 1. 0. 1. 1. 1. 0. 0. 0. 1. 0. 0. 1. 1.
 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.
 0. 1. 1. 0. 0. 0. 1. 1. 0. 1. 0. 1. 1. 0. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 1.
 1. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 0. 1. 0. 1. 0. 0. 0. 1. 0.
 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 0. 1. 0. 0. 1. 0. 1. 1. 1. 1.
 0. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 1.
 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 1. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 1.
 0. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 1. 0. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 1. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 1. 1. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 0.
 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1.
 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 0.]
```

In [ ]:

```python
#importing support vector classifier
from sklearn.svm import SVC
```

In [ ]:

```python
clf=SVC(kernel='linear') #choosing linear classifier
clf.fit(x,y)    #training
```

Out[ ]:

SVC

```
SVC(kernel='linear')
```

In [ ]:

```python
clf.predict([[120, 990]])
```

Out[ ]:

```
array([1.])
```

In [ ]:

```python
clf.predict([[85, 550]])
```

Out[ ]:

```
array([0.])
```

In [ ]:

```python
def make_meshgrid(x, y, h=0.2):#0.02
    x_min, x_max = x.min() - 1, x.max() + 1
    '''calculates the minimum and maximum values in the x array
    and adds/subtracts 1 to create a slightly extended range.
    This ensures the entire data is covered in the grid.'''
    y_min, y_max = y.min() - 1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    '''arrange:create an array of evenly spaced values starting
    from x_min (inclusive) and ending at x_max (exclusive) with a step size of h
    ''' takes two one-dimensional arrays and generates a two-dimensional grid of
    the first argument is the array of x-coord, and second argument is the array
    The output xx (containing the x-coord of the grid) and yy (containing the y-
    return xx, yy

def plot_contours(ax, clf, xx, yy, **params): #for plotting the contours
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, **params)
    return out
'''ax: a Matplotlib axis object where the plot will be created.
xx, yy: which are the two-dimensional grids of x and y coordinates
**params: This argument uses the double asterisk notation, indicating it can acc
 of keyword arguments. These arguments will be used to customize the contour plo
clf.predict(...): This method likely takes the combined (flattened) x and y coor
predicts a value for each data point based on the classifier's model.

Z.reshape(xx.shape): Since xx and yy represent the grid structure, this line res
output Z back into a two-dimensional array with the same shape as the original g
This ensures the predicted values are correctly mapped to the corresponding poin

ax.contourf(...): This line utilizes the contourf method of the Matplotlib axis
xx, yy: These are the two-dimensional grids containing the x and y coordinates f
Z: This is the two-dimensional array containing the predicted values at each gri
This array determines the height or elevation for the contour plot.
**params: The variable keyword arguments (**params) passed to the function are u
These might include options like the number of contour lines, colormap, line sty
out: This likely represents a Matplotlib contour object containing information a



The function returns the out variable, which holds the Matplotlib contour object
 This allows you to potentially interact with the plot object further (e.g., add
'''
```

Out[ ]: "\nax: a Matplotlib axis object where the plot will be created.\nxx, yy: which are the two-dimensional grids of x and y coordinates \n**params: This argument uses the double asterisk notation, indicating it can accept a variable number\nof keyword arguments. These arguments will be used to customize the contour plot.\nclf.predict(...): This method likely takes the combined (flattened) x and y coordinates from the grids (xx.ravel() and yy.ravel())\npredicts a value for each data point based on the classifier's model.\n\nZ.reshape(xx.shape): Since xx and yy represent the grid structure, this line reshapes the one-dimensional prediction \noutput Z back into a two-dimensional array with the same shape as the original grids xx and yy. \nThis ensures the predicted values are correctly mapped to the corresponding points in the grid.\n\nax.contourf(...): This line utilizes the contourf method of the Matplotlib axis object (ax).\nxx, yy: These are the two-dimensional grids containing the x and y coordinates for plotting the contours.\nZ: This is the two-dimensional array containing the predicted values at each grid point. \nThis array determines the height or elevation for the contour plot.\n**params: The variable keyword arguments (**params) passed to the function are used to customize the appearance of the contour plot. \nThese might include options like the number of contour lines, colormap, line styles, etc. \nout: This likely represents a Matplotlib contour object containing information about the plotted contours.\n\n\nThe function returns the out variable, which holds the Matplotlib contour object representing the generated filled contour plot.\n This allows you to potentially interact with the plot object further (e.g., adding labels, adjusting colors).\n"

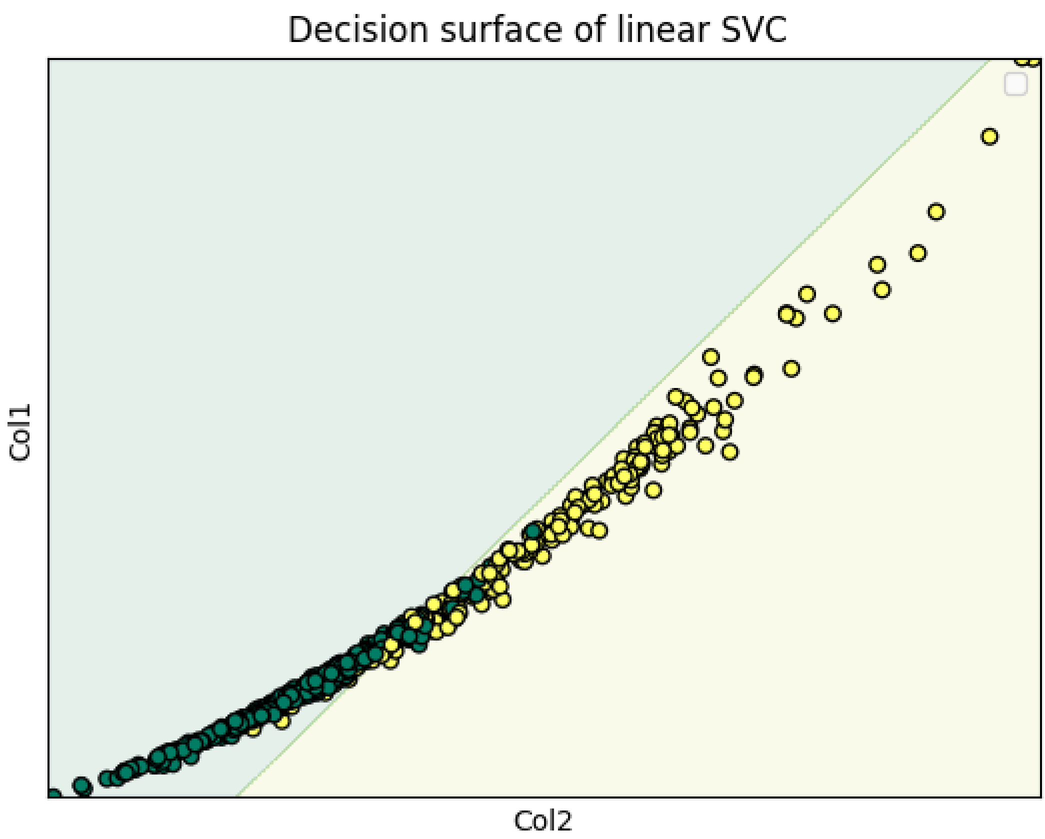
2.Run Linear and Poly

In [ ]:
```
X=x
model = SVC(kernel='linear')
clf = model.fit(X, y)
print(clf)
fig, ax = plt.subplots()
# title for the plots
title = ('Decision surface of linear SVC ')
# Set-up grid for plotting.
# print(X)
# print(y)
# print(X.iloc[:, 1])
X0, X1 = X[:, 0], X[:, 1]
print(X0,X1)

xx, yy = make_meshgrid(X0, X1)

plot_contours(ax, clf, xx, yy, cmap=plt.cm.summer, alpha=0.1)
ax.scatter(X0, X1, c=y, cmap=plt.cm.summer, s=30, edgecolors='k')
ax.set_ylabel('Col1')
ax.set_xlabel('Col2')
ax.set_xticks(())
ax.set_yticks(())
ax.set_title(title)
ax.legend()
plt.show()
```

```
SVC(kernel='linear')
[122.8  132.9  130.    77.58 135.1   82.57 119.6   90.2   87.5   83.97
 102.7  103.6  132.4  103.7   93.6   96.73  94.74 108.1  130.    87.46
  85.63  60.34 102.5  137.2  110.   116.    97.41 122.1  102.4  115.
 124.8   77.93 112.8  127.9  107.   110.1   93.63  82.61  95.54  88.4
  86.18  71.9  128.3   87.32  85.42 123.7   51.71  85.98  78.04  86.91
  74.72  87.21  75.71 120.3   97.26  73.34 125.5   95.55  82.61  54.34
  64.55  54.66  96.42  59.2   82.69  97.4   60.11  71.8   58.79  81.37
 123.6   58.79 114.2   90.43  79.19 104.1   87.91 120.2  143.7   83.19
  73.81  86.49 171.5  129.1   76.95 121.1   94.25 122.    79.78  95.77
  94.57 100.2   84.74  86.6  100.3  132.4   77.79  62.11  74.34  94.48
  88.05  43.79  77.22  63.95  67.41  87.21  75.17  79.01 152.8   72.48
  62.5   82.15  97.83  68.64  55.84  76.53  58.74  98.64 105.7  114.2
  73.34 121.4  166.2   94.28  86.1   88.44  87.76 123.4   99.58 130.4
  79.08 101.7  106.2  102.   120.2   81.72  74.72  73.06  96.85  73.
  61.24 105.1   73.66  83.74  68.26  78.11  78.99  97.84  93.97  88.12
  83.51  53.27  63.78  70.87  85.31  78.27 117.4  108.4   76.84  68.69
  76.1  126.3  130.7   79.85 152.1   95.5   68.77 109.3  116.1   96.22
  78.85  85.84 102.5   70.21  67.49  54.42  64.6  109.3   82.01  81.29
 182.1  142.7  101.2   73.53  98.92  63.76 118.6   74.68  75.27  78.83
  94.37  82.02  60.73  81.15 100.4   82.53  90.63 117.4  127.5   94.49
  78.54 115.1  158.9   91.56  81.09  98.78  62.92 109.7   87.02  98.17
 134.7   75.51 188.5  114.5   92.87  90.96  77.32  65.05 129.7  128.
  87.88  88.59  65.12 102.6   84.55  92.51  66.62  97.45  81.35  85.26
 113.4   71.76  70.79 134.4   60.21  89.79 153.5  132.5   92.55 113.4
  87.38  78.61  73.93  88.54 129.1   66.72  84.13  84.95  68.01  73.87
 138.9   73.28 130.7  113.   126.5   91.43 133.6  103.2  110.2  103.7
 132.9  111.   114.4  100.   111.6  135.7   69.28  87.16  82.38  69.5
  90.3   72.23 147.3   61.5  115.2   76.2   71.79 120.9   86.24  88.99
 126.2   74.24 127.2  108.8   84.08  79.83  77.87  81.89  73.72  72.17
  96.03  97.03  83.14  75.54  81.78  88.06  69.14  75.    91.22  66.85
 129.5   80.43 134.7   66.86  73.59  74.23  84.07  56.36  85.69  82.71
  74.33  92.68  82.29  73.73  54.09  79.19  77.25 118.7   60.07  78.6
  66.52 131.1   82.82 135.9   78.01  81.25  90.03  76.09 106.9  107.5
 105.8   84.52  71.94  71.38  77.88 111.8   84.08 122.9   64.41 155.1
  94.15  61.64  71.49 129.9   75.03  66.2   76.66  94.87  73.02  77.23
  73.7  107.1  174.2   98.    71.24  81.92  85.09  88.52  56.74  59.82
  79.42  85.24  81.87 106.6   85.48 133.8  133.7   78.31 140.9  147.2
 109.    97.65 141.3  134.8   87.84 106.3   70.15  85.89  88.27  73.3
  73.16  70.67  78.75  80.64  85.79  93.97  78.78  88.37  73.38 128.9
  65.75  55.27 102.4  144.4   78.07  89.75  88.1   83.05  70.31  75.26
 124.4   76.14  84.18  83.18  78.29  70.39 104.3   82.63 117.8   78.41
  72.49  70.92  59.75  97.53  96.71  76.39  59.6  102.9   80.88  70.95
  74.2   98.22  75.46  89.46  61.93  63.19  67.49  68.79  70.47  80.98
 102.1   81.47 133.8  123.7   94.89  91.12  82.67  89.78  88.68  89.59
  71.73 112.4   88.37  66.82 117.5   77.61 117.3   95.88  94.25 138.1
  76.83 127.7   76.77  93.86  80.62  86.34  74.87  84.1   82.61  61.68
 111.2  186.9   92.25  73.88  84.28  86.87  85.98  61.06 119.    76.38
  61.49  76.85  96.45  77.42  70.41  82.89  92.41  88.97  73.99 109.8
  78.29  88.73  87.32  87.76 102.8   82.85  94.21 128.1   75.49 107.1
  78.18 114.6  118.4   78.83  84.06  96.12  82.69  80.45 121.3  137.8
  98.73  92.33  81.25 152.1   61.49  64.12  79.47  71.25 104.7  103.8
  76.31  94.66  88.64  94.29  97.26  72.76 120.8  130.5   84.45  82.51
  59.96 165.5   71.3   88.73  63.    54.53  87.44  78.94  90.31  77.83
  75.89  75.21  87.76 134.7   70.79 137.8   93.77  76.37  47.98  48.34
  74.65  95.81  94.7   84.88  89.77  87.19  65.31  65.85  61.05  68.89
  68.51  71.49  81.35  59.01  82.5   65.67  64.73  59.26  96.39  74.52
  91.38  70.67 103.4  143.   142.   131.2  108.3  140.1   47.92] [1001.  1326.  1
203.   386.1 1297.   477.1 1040.   577.9  519.8  475.9
  797.8  781.  1123.   782.7  578.3  658.8  684.5  798.8 1260.   566.3
```

```
  520.    273.9  704.4 1404.    904.6  912.7  644.8 1094.    732.4  955.1
 1088.    440.6  899.3 1162.    807.2  869.5  633.    523.8  698.8  559.2
  563.    371.1 1104.    545.2  531.5 1076.    201.9  534.6  449.3  561.
  427.9  571.8  437.6 1033.    712.8  409.  1152.    656.9  527.2  224.5
  311.9  221.8  645.7  260.9  499.   668.3  269.4  394.1  250.5  502.5
 1130.    244.   929.4  584.1  470.9  817.7  559.2 1006.  1245.    506.3
  401.5  520.  1878.  1132.    443.3 1075.    648.2 1076.    466.1  651.9
  662.7  728.2  551.7  555.1  705.6 1264.    451.1  294.5  412.6  642.5
  582.7  143.5  458.7  298.3  336.1  530.2  412.5  466.7 1509.    396.5
  290.2  480.4  629.9  334.2  230.9  438.6  245.2  682.5  782.6  982.
  403.3 1077.  1761.    640.7  553.5  588.7  572.6 1138.    674.5 1192.
  455.8  748.9  809.8  761.7 1075.    506.3  423.6  399.8  678.1  384.8
  288.5  813.   398.   512.2  355.3  432.8  432.   689.5  640.1  585.
  519.4  203.9  300.2  381.9  538.9  460.3  963.7  880.2  448.6  366.8
  419.8 1157.  1214.    464.5 1686.    690.2  357.6  886.3  984.6  685.9
  464.1  565.4  736.9  372.7  349.6  227.2  302.4  832.9  526.4  508.8
 2250.  1311.    766.6  402.   710.6  317.5 1041.    420.3  428.9  463.7
  609.9  507.4  288.1  477.4  671.4  516.4  588.9 1024.  1148.    642.7
  461.   951.6 1685.    597.8  481.9  716.6  295.4  904.3  529.4  725.5
 1290.    428.  2499.    948.   610.7  578.9  432.2  321.2 1230.  1223.
  568.9  561.3  313.1  761.3  546.4  641.2  329.6  684.5  496.4  503.2
  895.   395.7  386.8 1319.    279.6  603.4 1670.  1306.    623.9  920.6
  575.3  476.5  389.4  590.  1155.    337.7  541.6  512.2  347.   406.3
 1364.    407.4 1206.    928.2 1169.    602.4 1207.    713.3  773.5  744.9
 1288.    933.1  947.8  758.6  928.3 1419.    346.4  561.   512.2  344.9
  632.6  388.  1491.    289.9  998.9  435.6  396.6 1102.    572.3  587.4
 1138.    427.3 1145.    805.1  516.6  489.   441.   515.9  394.1  396.
  651.   687.3  513.7  432.7  492.1  582.7  363.7  431.1  633.1  334.2
 1217.    471.3 1247.    334.3  403.1  417.2  537.3  246.3  566.2  530.6
  418.7  664.9  504.1  409.1  221.2  481.6  461.4 1027.    244.5  477.3
  324.2 1274.    504.8 1264.    457.9  489.9  616.5  446.   813.7  826.8
  793.2  514.   387.3  390.   464.4  918.6  514.3 1092.    310.8 1747.
  641.2  280.5  373.9 1194.    420.3  321.6  445.3  668.7  402.7  426.7
  421.   758.6 2010.    716.6  384.6  485.8  512.   593.7  241.   278.6
  491.9  546.1  496.6  838.1  552.4 1293.  1234.    458.4 1546.  1482.
  840.4  711.8 1386.  1335.    579.1  788.5  338.3  562.1  580.6  361.6
  386.3  372.7  447.8  462.9  541.8  664.7  462.   596.6  392.  1174.
  321.6  234.3  744.7 1407.    446.2  609.1  558.1  508.3  378.2  431.9
  994.   442.7  525.2  507.6  469.1  370.   800.   514.5  991.7  466.1
  399.8  373.2  268.8  693.7  719.5  433.8  271.2  803.1  495.   380.3
  409.7  656.1  408.2  575.3  289.7  307.3  333.6  359.9  381.1  501.3
  685.   467.8 1250.  1110.    673.7  599.5  509.2  611.2  592.6  606.5
  371.5  928.8  585.9  340.9  990.   441.3  981.6  674.8  659.7 1384.
  432.  1191.    442.5  644.2  492.9  557.2  415.1  537.9  520.2  290.9
  930.9 2501.    646.1  412.7  537.3  542.9  536.9  286.3  980.5  408.8
  289.1  449.9  686.9  465.4  358.9  506.9  618.4  599.4  404.9  815.8
  455.3  602.9  546.3  571.1  747.2  476.7  666.  1167.    420.5  857.6
  466.5  992.1 1007.    477.3  538.7  680.9  485.6  480.1 1068.  1320.
  689.4  595.9  476.3 1682.    248.7  272.5  453.1  366.5  819.8  731.3
  426.   680.7  556.7  658.8  701.9  391.2 1052.  1214.    493.1  493.8
  257.8 1841.    388.1  571.   293.2  221.3  551.1  468.5  594.2  445.2
  422.9  416.2  575.5 1299.    365.6 1308.    629.8  406.4  178.8  170.4
  402.9  656.4  668.6  538.4  584.8  573.2  324.9  320.8  285.7  361.6
  360.5  378.4  507.9  264.   514.3  321.4  311.7  271.3  657.1  403.5
  600.4  386.   716.9 1347.  1479.  1261.    858.1 1265.    181. ]
```

```
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
```



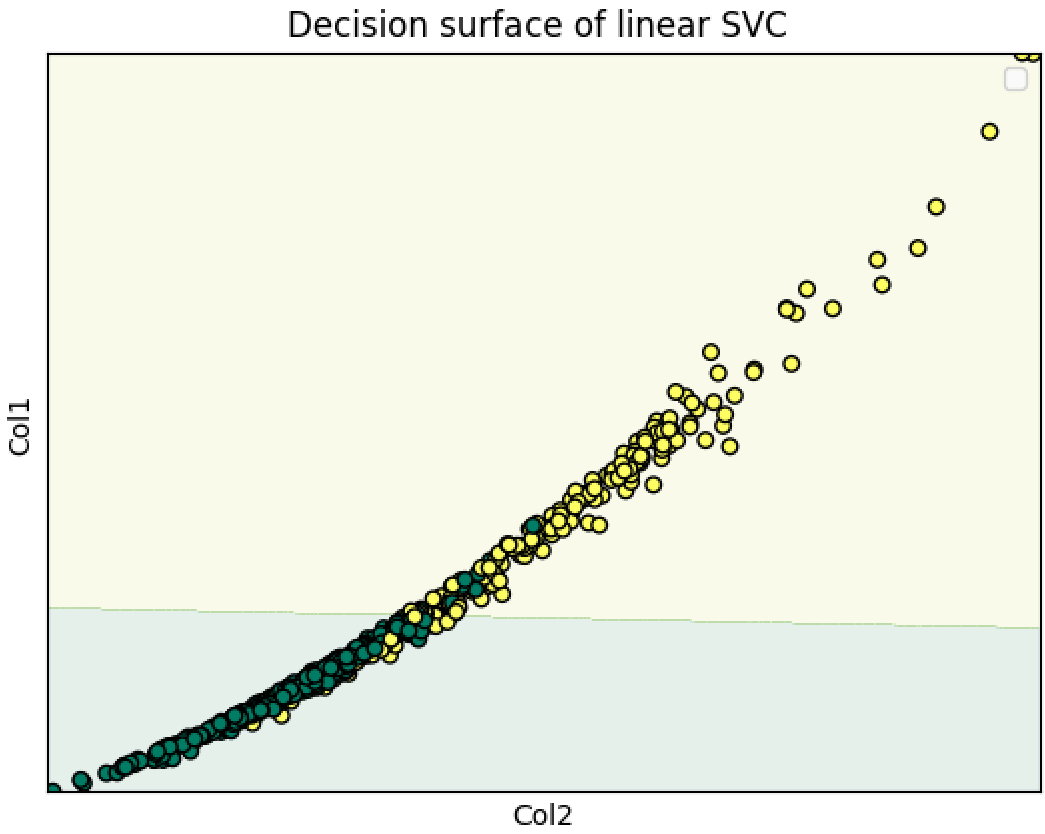
In [ ]:

```
X=x
model = SVC(kernel='poly')
clf = model.fit(X, y)
print(clf)
fig, ax = plt.subplots()
# title for the plots
title = ('Decision surface of linear SVC ')
# Set-up grid for plotting.
# print(X)
# print(y)
# print(X.iloc[:, 1])
X0, X1 = X[:, 0], X[:, 1]
print(X0,X1)

xx, yy = make_meshgrid(X0, X1)

plot_contours(ax, clf, xx, yy, cmap=plt.cm.summer, alpha=0.1)
ax.scatter(X0, X1, c=y, cmap=plt.cm.summer, s=30, edgecolors='k')
ax.set_ylabel('Col1')
ax.set_xlabel('Col2')
ax.set_xticks(())
ax.set_yticks(())
ax.set_title(title)
ax.legend()
plt.show()
```

```
SVC(kernel='poly')
[122.8  132.9  130.    77.58 135.1   82.57 119.6   90.2   87.5   83.97
 102.7  103.6  132.4  103.7   93.6   96.73  94.74 108.1  130.    87.46
  85.63  60.34 102.5  137.2  110.   116.    97.41 122.1  102.4  115.
 124.8   77.93 112.8  127.9  107.   110.1   93.63  82.61  95.54  88.4
  86.18  71.9  128.3   87.32  85.42 123.7   51.71  85.98  78.04  86.91
  74.72  87.21  75.71 120.3   97.26  73.34 125.5   95.55  82.61  54.34
  64.55  54.66  96.42  59.2   82.69  97.4   60.11  71.8   58.79  81.37
 123.6   58.79 114.2   90.43  79.19 104.1   87.91 120.2  143.7   83.19
  73.81  86.49 171.5  129.1   76.95 121.1   94.25 122.    79.78  95.77
  94.57 100.2   84.74  86.6  100.3  132.4   77.79  62.11  74.34  94.48
  88.05  43.79  77.22  63.95  67.41  87.21  75.17  79.01 152.8   72.48
  62.5   82.15  97.83  68.64  55.84  76.53  58.74  98.64 105.7  114.2
  73.34 121.4  166.2   94.28  86.1   88.44  87.76 123.4   99.58 130.4
  79.08 101.7  106.2  102.   120.2   81.72  74.72  73.06  96.85  73.
  61.24 105.1   73.66  83.74  68.26  78.11  78.99  97.84  93.97  88.12
  83.51  53.27  63.78  70.87  85.31  78.27 117.4  108.4   76.84  68.69
  76.1  126.3  130.7   79.85 152.1   95.5   68.77 109.3  116.1   96.22
  78.85  85.84 102.5   70.21  67.49  54.42  64.6  109.3   82.01  81.29
 182.1  142.7  101.2   73.53  98.92  63.76 118.6   74.68  75.27  78.83
  94.37  82.02  60.73  81.15 100.4   82.53  90.63 117.4  127.5   94.49
  78.54 115.1  158.9   91.56  81.09  98.78  62.92 109.7   87.02  98.17
 134.7   75.51 188.5  114.5   92.87  90.96  77.32  65.05 129.7  128.
  87.88  88.59  65.12 102.6   84.55  92.51  66.62  97.45  81.35  85.26
 113.4   71.76  70.79 134.4   60.21  89.79 153.5  132.5   92.55 113.4
  87.38  78.61  73.93  88.54 129.1   66.72  84.13  84.95  68.01  73.87
 138.9   73.28 130.7  113.   126.5   91.43 133.6  103.2  110.2  103.7
 132.9  111.   114.4  100.   111.6  135.7   69.28  87.16  82.38  69.5
  90.3   72.23 147.3   61.5  115.2   76.2   71.79 120.9   86.24  88.99
 126.2   74.24 127.2  108.8   84.08  79.83  77.87  81.89  73.72  72.17
  96.03  97.03  83.14  75.54  81.78  88.06  69.14  75.    91.22  66.85
 129.5   80.43 134.7   66.86  73.59  74.23  84.07  56.36  85.69  82.71
  74.33  92.68  82.29  73.73  54.09  79.19  77.25 118.7   60.07  78.6
  66.52 131.1   82.82 135.9   78.01  81.25  90.03  76.09 106.9  107.5
 105.8   84.52  71.94  71.38  77.88 111.8   84.08 122.9   64.41 155.1
  94.15  61.64  71.49 129.9   75.03  66.2   76.66  94.87  73.02  77.23
  73.7  107.1  174.2   98.    71.24  81.92  85.09  88.52  56.74  59.82
  79.42  85.24  81.87 106.6   85.48 133.8  133.7   78.31 140.9  147.2
 109.    97.65 141.3  134.8   87.84 106.3   70.15  85.89  88.27  73.3
  73.16  70.67  78.75  80.64  85.79  93.97  78.78  88.37  73.38 128.9
  65.75  55.27 102.4  144.4   78.07  89.75  88.1   83.05  70.31  75.26
 124.4   76.14  84.18  83.18  78.29  70.39 104.3   82.63 117.8   78.41
  72.49  70.92  59.75  97.53  96.71  76.39  59.6  102.9   80.88  70.95
  74.2   98.22  75.46  89.46  61.93  63.19  67.49  68.79  70.47  80.98
 102.1   81.47 133.8  123.7   94.89  91.12  82.67  89.78  88.68  89.59
  71.73 112.4   88.37  66.82 117.5   77.61 117.3   95.88  94.25 138.1
  76.83 127.7   76.77  93.86  80.62  86.34  74.87  84.1   82.61  61.68
 111.2  186.9   92.25  73.88  84.28  86.87  85.98  61.06 119.    76.38
  61.49  76.85  96.45  77.42  70.41  82.89  92.41  88.97  73.99 109.8
  78.29  88.73  87.32  87.76 102.8   82.85  94.21 128.1   75.49 107.1
  78.18 114.6  118.4   78.83  84.06  96.12  82.69  80.45 121.3  137.8
  98.73  92.33  81.25 152.1   61.49  64.12  79.47  71.25 104.7  103.8
  76.31  94.66  88.64  94.29  97.26  72.76 120.8  130.5   84.45  82.51
  59.96 165.5   71.3   88.73  63.    54.53  87.44  78.94  90.31  77.83
  75.89  75.21  87.76 134.7   70.79 137.8   93.77  76.37  47.98  48.34
  74.65  95.81  94.7   84.88  89.77  87.19  65.31  65.85  61.05  68.89
  68.51  71.49  81.35  59.01  82.5   65.67  64.73  59.26  96.39  74.52
  91.38  70.67 103.4  143.   142.   131.2  108.3  140.1   47.92] [1001.  1326.  1
203.   386.1 1297.   477.1 1040.   577.9  519.8  475.9
  797.8  781.  1123.   782.7  578.3  658.8  684.5  798.8 1260.   566.3
```

```
  520.    273.9  704.4 1404.    904.6  912.7  644.8 1094.    732.4  955.1
 1088.    440.6  899.3 1162.    807.2  869.5  633.    523.8  698.8  559.2
  563.    371.1 1104.    545.2  531.5 1076.    201.9  534.6  449.3  561.
  427.9  571.8  437.6 1033.    712.8  409.  1152.    656.9  527.2  224.5
  311.9  221.8  645.7  260.9  499.    668.3  269.4  394.1  250.5  502.5
 1130.    244.    929.4  584.1  470.9  817.7  559.2 1006.   1245.    506.3
  401.5  520.   1878.   1132.    443.3 1075.    648.2 1076.    466.1  651.9
  662.7  728.2  551.7  555.1  705.6 1264.    451.1  294.5  412.6  642.5
  582.7  143.5  458.7  298.3  336.1  530.2  412.5  466.7 1509.    396.5
  290.2  480.4  629.9  334.2  230.9  438.6  245.2  682.5  782.6  982.
  403.3 1077.   1761.    640.7  553.5  588.7  572.6 1138.    674.5 1192.
  455.8  748.9  809.8  761.7 1075.    506.3  423.6  399.8  678.1  384.8
  288.5  813.    398.    512.2  355.3  432.8  432.    689.5  640.1  585.
  519.4  203.9  300.2  381.9  538.9  460.3  963.7  880.2  448.6  366.8
  419.8 1157.   1214.    464.5 1686.    690.2  357.6  886.3  984.6  685.9
  464.1  565.4  736.9  372.7  349.6  227.2  302.4  832.9  526.4  508.8
 2250.   1311.    766.6  402.    710.6  317.5 1041.    420.3  428.9  463.7
  609.9  507.4  288.1  477.4  671.4  516.4  588.9 1024.   1148.    642.7
  461.    951.6 1685.    597.8  481.9  716.6  295.4  904.3  529.4  725.5
 1290.    428.   2499.    948.    610.7  578.9  432.2  321.2 1230.   1223.
  568.9  561.3  313.1  761.3  546.4  641.2  329.6  684.5  496.4  503.2
  895.    395.7  386.8 1319.    279.6  603.4 1670.   1306.    623.9  920.6
  575.3  476.5  389.4  590.   1155.    337.7  541.6  512.2  347.    406.3
 1364.    407.4 1206.    928.2 1169.    602.4 1207.    713.3  773.5  744.9
 1288.    933.1  947.8  758.6  928.3 1419.    346.4  561.    512.2  344.9
  632.6  388.   1491.    289.9  998.9  435.6  396.6 1102.    572.3  587.4
 1138.    427.3 1145.    805.1  516.6  489.    441.    515.9  394.1  396.
  651.    687.3  513.7  432.7  492.1  582.7  363.7  431.1  633.1  334.2
 1217.    471.3 1247.    334.3  403.1  417.2  537.3  246.3  566.2  530.6
  418.7  664.9  504.1  409.1  221.2  481.6  461.4 1027.    244.5  477.3
  324.2 1274.    504.8 1264.    457.9  489.9  616.5  446.    813.7  826.8
  793.2  514.    387.3  390.    464.4  918.6  514.3 1092.    310.8 1747.
  641.2  280.5  373.9 1194.    420.3  321.6  445.3  668.7  402.7  426.7
  421.    758.6 2010.    716.6  384.6  485.8  512.    593.7  241.    278.6
  491.9  546.1  496.6  838.1  552.4 1293.   1234.    458.4 1546.   1482.
  840.4  711.8 1386.   1335.    579.1  788.5  338.3  562.1  580.6  361.6
  386.3  372.7  447.8  462.9  541.8  664.7  462.    596.6  392.   1174.
  321.6  234.3  744.7 1407.    446.2  609.1  558.1  508.3  378.2  431.9
  994.    442.7  525.2  507.6  469.1  370.    800.    514.5  991.7  466.1
  399.8  373.2  268.8  693.7  719.5  433.8  271.2  803.1  495.    380.3
  409.7  656.1  408.2  575.3  289.7  307.3  333.6  359.9  381.1  501.3
  685.    467.8 1250.   1110.    673.7  599.5  509.2  611.2  592.6  606.5
  371.5  928.8  585.9  340.9  990.    441.3  981.6  674.8  659.7 1384.
  432.   1191.    442.5  644.2  492.9  557.2  415.1  537.9  520.2  290.9
  930.9 2501.    646.1  412.7  537.3  542.9  536.9  286.3  980.5  408.8
  289.1  449.9  686.9  465.4  358.9  506.9  618.4  599.4  404.9  815.8
  455.3  602.9  546.3  571.1  747.2  476.7  666.   1167.    420.5  857.6
  466.5  992.1 1007.    477.3  538.7  680.9  485.6  480.1 1068.   1320.
  689.4  595.9  476.3 1682.    248.7  272.5  453.1  366.5  819.8  731.3
  426.    680.7  556.7  658.8  701.9  391.2 1052.   1214.    493.1  493.8
  257.8 1841.    388.1  571.    293.2  221.3  551.1  468.5  594.2  445.2
  422.9  416.2  575.5 1299.    365.6 1308.    629.8  406.4  178.8  170.4
  402.9  656.4  668.6  538.4  584.8  573.2  324.9  320.8  285.7  361.6
  360.5  378.4  507.9  264.    514.3  321.4  311.7  271.3  657.1  403.5
  600.4  386.    716.9 1347.   1479.   1261.    858.1 1265.    181. ]
```

```
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
```



3)Check accuracy in terms of percentage if possible try train test split to get training accuracy and testing accuracy

In [ ]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y) #for linear svm
clf=svm.SVC(kernel='linear').fit(x_train,y_train)
train_pred=clf.predict(x_train)
test_pred=clf.predict(x_test)
print("training accuracy score",accuracy_score(y_train, train_pred))
print("testing accuracy score: ",accuracy_score(y_test,test_pred))  #we probably
```

```
training accuracy score 0.8849765258215962
testing accuracy score:  0.9090909090909091
```

5.Check SVM with Data we used in Decision Tree

In [ ]:
```
df=pd.read_csv("dataTree1.csv")
d = {'UK': 0, 'USA': 1, 'N': 2}
df['Nationality'] = df['Nationality'].map(d)
d = {'YES': 1, 'NO': 0}
df['Go'] = df['Go'].map(d)
a = np.array(df)
y = a[:,4]  #gets all rows of 5th column
# extracting two features
x = np.column_stack((df.Age,df.Experience,df.Rank,df.Nationality))

X=x
Y=y
model=SVC(kernel='linear')
clf=model.fit(X,Y)
print(clf) #45,9,9,UK,YES   #24,3,5,USA,NO
```

```
SVC(kernel='linear')
```

In [ ]:
```
print(clf.predict([[24,3,5,1]]))#output is 0 i.e. NO
```

```
[0]
```

In [ ]:
```
print(clf.predict([[45,9,9,0]]))
```

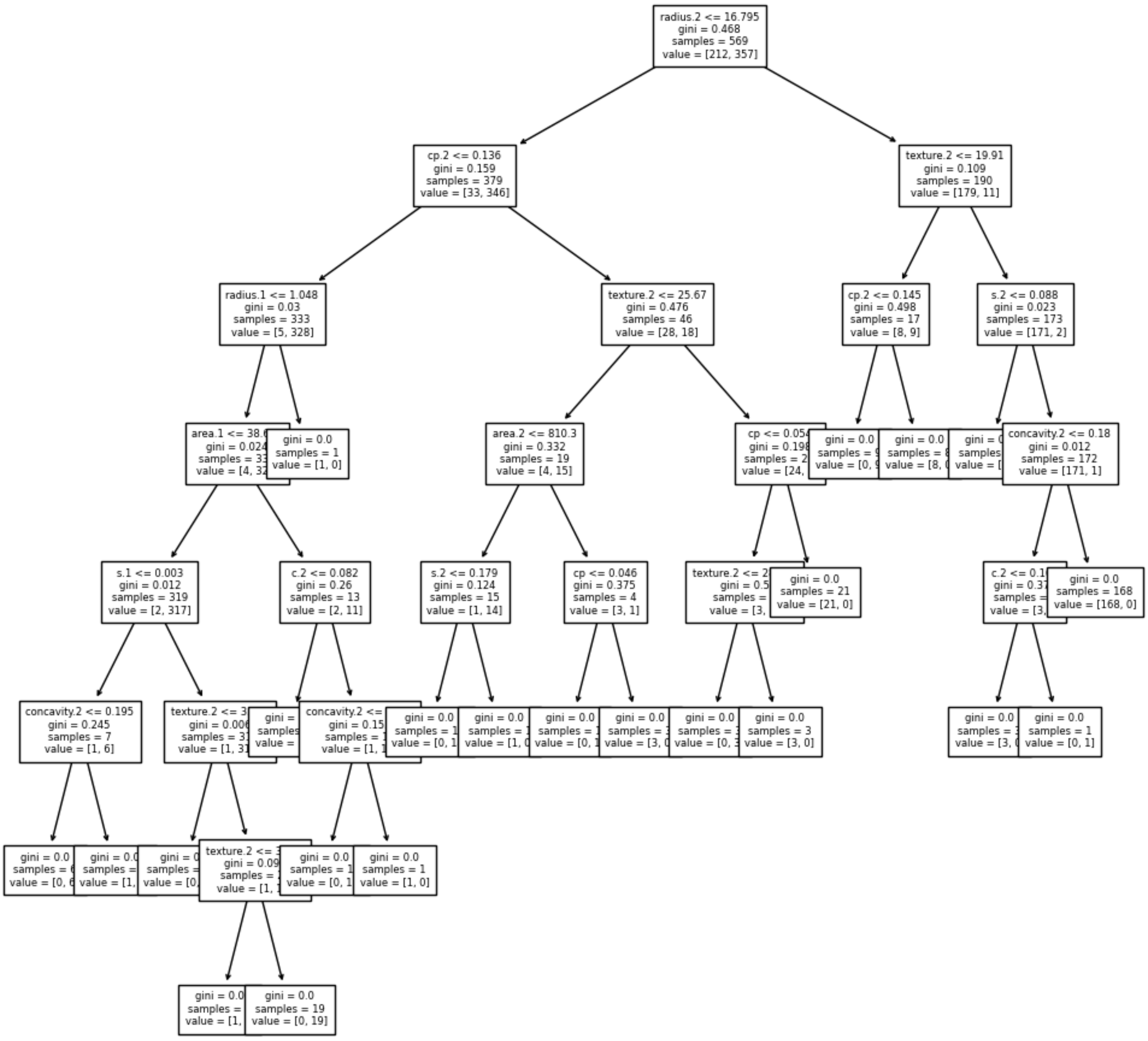
```
[1]
```

6. Run the Decision Tree for Cancer ALL data with 30 columns for X and Diagnosis column for y Please remove ID column

In [ ]:
```
df=pd.read_excel("cancerAll.xlsx")
map_df={"M":0,"B":1}
df["diagnosis"]=df["diagnosis"].map(map_df)
df.drop(columns=["ID"],inplace=True)
a=np.array(df)
y=a[:,0]
x=a[:,1:]
dtree=DecisionTreeClassifier()
dtree.fit(x,y)
features=list(df.columns)
features.remove("diagnosis")
plt.figure(figsize=(12,12))
tree.plot_tree(dtree,feature_names=features,fontsize=6)
```

```
Out[ ]: [Text(0.625, 0.9375, 'radius.2 <= 16.795\ngini = 0.468\nsamples = 569\nvalue =
        [212, 357]'),
         Text(0.40625, 0.8125, 'cp.2 <= 0.136\ngini = 0.159\nsamples = 379\nvalue = [3
        3, 346]'),
         Text(0.234375, 0.6875, 'radius.1 <= 1.048\ngini = 0.03\nsamples = 333\nvalue =
        [5, 328]'),
         Text(0.203125, 0.5625, 'area.1 <= 38.605\ngini = 0.024\nsamples = 332\nvalue =
        [4, 328]'),
         Text(0.125, 0.4375, 's.1 <= 0.003\ngini = 0.012\nsamples = 319\nvalue = [2, 31
        7]'),
         Text(0.0625, 0.3125, 'concavity.2 <= 0.195\ngini = 0.245\nsamples = 7\nvalue =
        [1, 6]'),
         Text(0.03125, 0.1875, 'gini = 0.0\nsamples = 6\nvalue = [0, 6]'),
         Text(0.09375, 0.1875, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
         Text(0.1875, 0.3125, 'texture.2 <= 33.27\ngini = 0.006\nsamples = 312\nvalue =
        [1, 311]'),
         Text(0.15625, 0.1875, 'gini = 0.0\nsamples = 292\nvalue = [0, 292]'),
         Text(0.21875, 0.1875, 'texture.2 <= 33.56\ngini = 0.095\nsamples = 20\nvalue =
        [1, 19]'),
         Text(0.1875, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
         Text(0.25, 0.0625, 'gini = 0.0\nsamples = 19\nvalue = [0, 19]'),
         Text(0.28125, 0.4375, 'c.2 <= 0.082\ngini = 0.26\nsamples = 13\nvalue = [2, 1
        1]'),
         Text(0.25, 0.3125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
         Text(0.3125, 0.3125, 'concavity.2 <= 0.328\ngini = 0.153\nsamples = 12\nvalue
        = [1, 11]'),
         Text(0.28125, 0.1875, 'gini = 0.0\nsamples = 11\nvalue = [0, 11]'),
         Text(0.34375, 0.1875, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
         Text(0.265625, 0.5625, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
         Text(0.578125, 0.6875, 'texture.2 <= 25.67\ngini = 0.476\nsamples = 46\nvalue
        = [28, 18]'),
         Text(0.46875, 0.5625, 'area.2 <= 810.3\ngini = 0.332\nsamples = 19\nvalue =
        [4, 15]'),
         Text(0.40625, 0.4375, 's.2 <= 0.179\ngini = 0.124\nsamples = 15\nvalue = [1, 1
        4]'),
         Text(0.375, 0.3125, 'gini = 0.0\nsamples = 14\nvalue = [0, 14]'),
         Text(0.4375, 0.3125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
         Text(0.53125, 0.4375, 'cp <= 0.046\ngini = 0.375\nsamples = 4\nvalue = [3,
        1]'),
         Text(0.5, 0.3125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
         Text(0.5625, 0.3125, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
         Text(0.6875, 0.5625, 'cp <= 0.054\ngini = 0.198\nsamples = 27\nvalue = [24,
        3]'),
         Text(0.65625, 0.4375, 'texture.2 <= 28.545\ngini = 0.5\nsamples = 6\nvalue =
        [3, 3]'),
         Text(0.625, 0.3125, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
         Text(0.6875, 0.3125, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
         Text(0.71875, 0.4375, 'gini = 0.0\nsamples = 21\nvalue = [21, 0]'),
         Text(0.84375, 0.8125, 'texture.2 <= 19.91\ngini = 0.109\nsamples = 190\nvalue
        = [179, 11]'),
         Text(0.78125, 0.6875, 'cp.2 <= 0.145\ngini = 0.498\nsamples = 17\nvalue = [8,
        9]'),
         Text(0.75, 0.5625, 'gini = 0.0\nsamples = 9\nvalue = [0, 9]'),
         Text(0.8125, 0.5625, 'gini = 0.0\nsamples = 8\nvalue = [8, 0]'),
         Text(0.90625, 0.6875, 's.2 <= 0.088\ngini = 0.023\nsamples = 173\nvalue = [17
        1, 2]'),
         Text(0.875, 0.5625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
         Text(0.9375, 0.5625, 'concavity.2 <= 0.18\ngini = 0.012\nsamples = 172\nvalue
        = [171, 1]'),
         Text(0.90625, 0.4375, 'c.2 <= 0.161\ngini = 0.375\nsamples = 4\nvalue = [3,
```

```
1]'),
 Text(0.875, 0.3125, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(0.9375, 0.3125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.96875, 0.4375, 'gini = 0.0\nsamples = 168\nvalue = [168, 0]')]
```



7. Identify main columns in Step 6 from decision tree made and run for those columns on SVM

The main columns are radius.2, cp.2, texture,sym.1, texture.1,s.2, area.1,area.2, concavity,concavity.2,s.1,sym.1,cp,cp.1, sym, texture.2,

In [ ]:
```
# removing all unwanted columns
df.drop(columns=['perimeter', 'area', 's', 'c',  'fd', 'radius.1', 'perimeter.1'
a=np.array(df)
y=a[:,0]
x=a[:,1:]
model=SVC(kernel='linear')
x_train,x_test,y_train,y_test=train_test_split(x,y) #for linear svm
model=SVC(kernel='linear').fit(x_train,y_train)
train_pred=clf.predict(x_train)
test_pred=clf.predict(x_test)
print("training accuracy score",accuracy_score(y_train, train_pred))
print("testing accuracy score: ",accuracy_score(y_test,test_pred))  #we probably
```

```
training accuracy score 0.9671361502347418
testing accuracy score:  0.9370629370629371
```