In [ ]:
```
#https://www.w3schools.com/python/python_ml_decision_tree.asp
```

Machine Learning Lab3: Created by Jibrael Jos,PhD

Topic: SVM Explorations

Student Name: Naveen Krishna

Roll No:23122023

Date: 15 March

Submission : 4th April

In [ ]:
```
import pandas

df = pandas.read_csv("cancerAllv3.csv")

print(df)
```

```
     radius  texture  perimeter    area        s        c  concavity
cp  \
0     17.99    10.38     122.80  1001.0  0.11840  0.27760    0.30010  0.14
710
1     20.57    17.77     132.90  1326.0  0.08474  0.07864    0.08690  0.07
017
2     19.69    21.25     130.00  1203.0  0.10960  0.15990    0.19740  0.12
790
3     11.42    20.38      77.58   386.1  0.14250  0.28390    0.24140  0.10
520
4     20.29    14.34     135.10  1297.0  0.10030  0.13280    0.19800  0.10
430
..      ...      ...        ...     ...      ...      ...        ...
...
564   21.56    22.39     142.00  1479.0  0.11100  0.11590    0.24390  0.13
890
565   20.13    28.25     131.20  1261.0  0.09780  0.10340    0.14400  0.09
791
566   16.60    28.08     108.30   858.1  0.08455  0.10230    0.09251  0.05
302
567   20.60    29.33     140.10  1265.0  0.11780  0.27700    0.35140  0.15
200
568    7.76    24.54      47.92   181.0  0.05263  0.04362    0.00000  0.00
000

        sym       fd  ...  texture2  perimeter2   area2       s2       c2
\
0    0.2419  0.07871  ...     17.33      184.60  2019.0  0.16220  0.66560
1    0.1812  0.05667  ...     23.41      158.80  1956.0  0.12380  0.18660
2    0.2069  0.05999  ...     25.53      152.50  1709.0  0.14440  0.42450
3    0.2597  0.09744  ...     26.50       98.87   567.7  0.20980  0.86630
4    0.1809  0.05883  ...     16.67      152.20  1575.0  0.13740  0.20500
..      ...      ...  ...       ...         ...     ...      ...      ...
564  0.1726  0.05623  ...     26.40      166.10  2027.0  0.14100  0.21130
565  0.1752  0.05533  ...     38.25      155.00  1731.0  0.11660  0.19220
566  0.1590  0.05648  ...     34.12      126.70  1124.0  0.11390  0.30940
567  0.2397  0.07016  ...     39.42      184.60  1821.0  0.16500  0.86810
568  0.1587  0.05884  ...     30.37       59.16   268.6  0.08996  0.06444

     concavity2     cp2    sym2      fd2  diagnosis
0        0.7119  0.2654  0.4601  0.11890          1
1        0.2416  0.1860  0.2750  0.08902          1
2        0.4504  0.2430  0.3613  0.08758          1
3        0.6869  0.2575  0.6638  0.17300          1
4        0.4000  0.1625  0.2364  0.07678          1
..          ...     ...     ...      ...        ...
564      0.4107  0.2216  0.2060  0.07115          1
565      0.3215  0.1628  0.2572  0.06637          1
566      0.3403  0.1418  0.2218  0.07820          1
567      0.9387  0.2650  0.4087  0.12400          1
568      0.0000  0.0000  0.2871  0.07039          0

[569 rows x 31 columns]
```

In [ ]:
```
features=['radius','texture','perimeter','area','s','c','concavity','cp',

import numpy as np
X = np.array(df)
y = X[:,30]
X = X[:,0:9]
```

```
print(X)
print(y)
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 3.001e-01 1.471e-01 2.419e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 8.690e-02 7.017e-02 1.812e-01]
 [1.969e+01 2.125e+01 1.300e+02 ... 1.974e-01 1.279e-01 2.069e-01]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 9.251e-02 5.302e-02 1.590e-01]
 [2.060e+01 2.933e+01 1.401e+02 ... 3.514e-01 1.520e-01 2.397e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 0.000e+00 1.587e-01]]
[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. 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. 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 [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y ,
                                    random_state=104,
                                    test_size=0.25,
                                    shuffle=True)
```

In [ ]:
```
import pandas
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt

from sklearn.svm import SVC
clf = SVC(kernel='linear')

# fitting x samples and y classes
clf.fit(X_train, y_train)
```

Out[ ]:
```
▾        SVC

SVC(kernel='linear')
```

Accuracy for training dataset.

In [ ]:
```
from sklearn.metrics import accuracy_score
predicted = clf.predict(X_train)
print (accuracy_score(y_train, predicted))
```

```
0.9225352112676056
```

Accuracy for testing dataset.

In [ ]:
```
from sklearn.metrics import accuracy_score
predicted = clf.predict(X_test)
print (accuracy_score(y_test, predicted))
```

```
0.9020979020979021
```

Classification Report

In [ ]:
```
from sklearn.metrics import classification_report
features=['radius','texture','perimeter','area','s','c','concavity','cp',
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.91      0.93      0.92        87
         1.0       0.89      0.86      0.87        56

    accuracy                           0.90       143
   macro avg       0.90      0.89      0.90       143
weighted avg       0.90      0.90      0.90       143
```

Confussion Matrix

In [ ]:
```
from sklearn.metrics import confusion_matrix



conf_matrix = confusion_matrix(y_test, y_pred)

print( conf_matrix)
```

```
[[81  6]
 [ 8 48]]
```

Grid Search

In [ ]:
```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC


# Define the parameter grid to search
param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': [ 'linear', 'poly']
}
```

```
# Create a grid search object
grid_search = GridSearchCV(SVC(), param_grid, cv=5)

# Perform grid search on the training data
grid_search.fit(X_train, y_train)

# Get the best hyperparameters and model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

# Evaluate the best model on the test set
accuracy = best_model.score(X_test, y_test)

print("Best hyperparameters:", best_params)
print("Test set accuracy:", accuracy)
```

```
Best hyperparameters: {'C': 10, 'kernel': 'linear'}
Test set accuracy: 0.9090909090909091
```