**Practical – 4: Application of Decision Tree and SVM**

**2) SVM**

**Description:** In this practical, we will use SVM (Support Vector Machines) to build and train a model using human cell records, and classify cells to whether the samples are benign or malignant.

The dataset consists of several hundred human cell sample records, each of which contains the values of a set of cell characteristics. The fields in each record are:



**Code:**

**# import file and load data set**

import pandas as pd

import pylab as pl

import numpy as np

import scipy.optimize as opt

from sklearn import preprocessing

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

cell\_df = pd.read\_csv("cell\_samples.csv")

cell\_df.head()

ax = cell\_df[cell\_df['Class'] == 4][0:50].plot(kind='scatter', x='Clump', y='UnifSize', color='DarkBlue', label='malignant');

cell\_df[cell\_df['Class'] == 2][0:50].plot(kind='scatter', x='Clump', y='UnifSize', color='Yellow', label='benign', ax=ax);

plt.show()

print(cell\_df.dtypes)

# it looks like the BareNuc column includes some values that are not numerical.

# We can drop those rows:

cell\_df = cell\_df[pd.to\_numeric(cell\_df['BareNuc'], errors='coerce').notnull()]

cell\_df['BareNuc'] = cell\_df['BareNuc'].astype('int')

cell\_df.dtypes

feature\_df = cell\_df[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize', 'BareNuc', 'BlandChrom', 'NormNucl', 'Mit']]

X = np.asarray(feature\_df)

X[0:5]

# We want the model to predict the value of Class (that is, benign (=2) or malignant (=4)).

# As this field can have one of only two possible values,

# we need to change its measurement level to reflect this.

cell\_df['Class'] = cell\_df['Class'].astype('int')

y = np.asarray(cell\_df['Class'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.2, random\_state=4)

print ('Train set:', X\_train.shape, y\_train.shape)

print ('Test set:', X\_test.shape, y\_test.shape)

**# fitting model**

from sklearn import svm

clf = svm.SVC(kernel='rbf')

clf.fit(X\_train, y\_train)

# After being fitted, the model can then be used to predict new values:

yhat = clf.predict(X\_test)

print("predicted Y: ", yhat [0:5])

**# Evaluation**

from sklearn.metrics import f1\_score

print("Accuracy : ", f1\_score(y\_test, yhat, average='weighted') )

**Output:**

ID int64

Clump int64

UnifSize int64

UnifShape int64

MargAdh int64

SingEpiSize int64

BareNuc object

BlandChrom int64

NormNucl int64

Mit int64

Class int64

Train set: (546, 9) (546,)

Test set: (137, 9) (137,)

predicted Y: [2 4 2 4 2]

Accuracy : 0.9639038982104676