# #dataset=data.csv downloaded from Breast Cancer Wisconsin (Diagnostic) Data Set

Attribute Information:

1. ID number 2) Diagnosis (M = malignant, B = benign) 3–32)

Ten real-valued features are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter² / area — 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension (“coastline approximation” — 1)

The mean, standard error and “worst” or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

prediction\_var = ['texture\_mean','perimeter\_mean','smoothness\_mean','compactness\_mean','symmetry\_mean']

Based on above features prediction is done.

Program

Import matplotlib.pyplot as plt#used for plotting graph

from sklearn.model\_selection import train\_test\_split# to split the data into two parts

fromsklearn.ensemble import RandomForestClassifier#forrandomforest classifier

fromsklearn.metrics import accuracy\_score# to check the error and accuracy of the model

fromsklearn import svm#for support vector machine

import pandas as pd # data processing, CSV file I/O

import seaborn as sns # for statistical data visualization.

fromIPython import get\_ipython

get\_ipython().run\_line\_magic('matplotlib', 'inline')

data = pd.read\_csv("C:/Users/admin/.spyder-py3/data.csv",header=0)# here header 0 means the 0 th row is the coloumn

# header in data

print(data.head(2))# The data have imported and having 33 columns

# head is used for to see top 5 by default . Since head(2) is given it will print top 2 rows

data.info()

# drop the column Unnamed: 32

data.drop("Unnamed: 32",axis=1,inplace=True) # for dropping or the coloumn Unnamed: 32

# axis 1 means droping the column

data.columns# gives the column name which are persent in the data set

# To remove Id column for analysis

data.drop("id",axis=1,inplace=True)

# The data is divided into three parts. Divided the features according to their category

features\_mean= list(data.columns[1:11])

features\_se= list(data.columns[11:20])

features\_worst=list(data.columns[21:31])

print(features\_mean)

print("-----------------------------------")

print(features\_se)

print("------------------------------------")

print(features\_worst)

# diagnosis column is a object type , map it to integer value

data['diagnosis']=data['diagnosis'].map({'M':1,'B':0})

#data.describe() # this will describe the all statistical function of our data

# get the frequency of cancer stages

sns.countplot(data['diagnosis'],label="Count")

# draw a correlation graph so that , can remove multi colinearity

corr = data[features\_mean].corr() # .corr is used for find corelation

plt.figure(figsize=(14,14))

sns.heatmap(corr, cbar = True, square = True, annot=True, fmt= '.2f',annot\_kws={'size': 15},

xticklabels= features\_mean, yticklabels= features\_mean,

cmap= 'coolwarm')

prediction\_var = ['texture\_mean','perimeter\_mean','smoothness\_mean','compactness\_mean','symmetry\_mean']

# now these are the variables which will use for prediction

#now split the data into train and test

train, test = train\_test\_split(data, test\_size = 0.3)# in this our main data is splitted into train and test

# we can check their dimension

print(train.shape)

print(test.shape)

train\_X = train[prediction\_var]# taking the training data input

train\_y=train.diagnosis# This is output of our training data

# same to do for test

test\_X= test[prediction\_var] # taking test data inputs

test\_y =test.diagnosis #output value of test dat

model=RandomForestClassifier(n\_estimators=100)# a simple random forest model

model.fit(train\_X,train\_y)# now fit the model for traiing data

prediction=model.predict(test\_X)# predict for the test data

# prediction will contain the predicted value by our model predicted values of dignosis column for test inputs

print ("accuracy of randomforest")

print (accuracy\_score(prediction,test\_y))# to check the accuracy

# here will use accuracy measurement between the predicted value and the test output values

model = svm.SVC()

model.fit(train\_X,train\_y)

prediction=model.predict(test\_X)

print ("accuracy of SVM")

print (accuracy\_score(prediction,test\_y))