**Experiment No. – 7 K-Nearest Neighbour Classifier**

* Implement a K-Nearest Neighbors classifier on a breast cancer dataset to predict whether a tumor is malignant (cancerous) or benign (non-cancerous).
* Use given data for learning purposes. Then take new data for classification and perform experiment.

**Importing libraries and Reading Data - Code**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

df=pd.read\_csv("breast\_cancer.csv")

df.head()

**Remove unwanted columns:**

df.drop(['id', 'Unnamed: 32'], axis = 1,inplace=True)

**check whether any of the columns contain null values**

**df.isnull().sum()**

The label values are ‘M’ and ‘B’ corresponding to the Malignant and Benign classes. We can convert them to 0 and 1 respectively.

**ctypes ={'M' : 1, 'B' : 0}**

**df['diagnosis'] = df['diagnosis'].map(ctypes)**

**Visualizing the Data:**

**sns.pairplot(df,vars=['radius\_mean','texture\_mean','perimeter\_mean'],hue='diagnosis')**

**Creating the KNN Model:**

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

**from sklearn.neighbors import KNeighborsClassifier**

**Before feeding the data to the algorithm, we split the data into labels and features.**

**X = np.array(df.iloc[:,1:])**

**y = np.array(df['diagnosis'])**

**For evaluating the model we have to take train and test datasets separately.**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size = 0.33, random\_state = 42)**

**Creating the Model with selected optimal value:**

**knn = KNeighborsClassifier(n\_neighbors = 13)**

**knn.fit(X\_train,y\_train)**

**The model created here is:**

**KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',**

**metric\_params=None, n\_jobs=None, n\_neighbors=13, p=2, weights='uniform')**

**Evaluating the Model**

**from sklearn.metrics import accuracy\_score,classification\_report,confusion\_matrix**

**y\_predict=knn.predict(X\_test)**

**#accuracy=knn.score(X\_test,y\_test)**

**#print(accuracy)**

**acc=accuracy\_score(y\_test,y\_predict)**

**print(f'Accuracy Score of the Model: {acc}')**

**//Accuracy Score of the Model: 0.9627659574468085**

**Confusion Matrix**

**print('Confusion Matrix :\n')**

**print(confusion\_matrix(y\_test,y\_predict))**

**print(f'\nClassification Report \n\n {classification\_report(y\_test,y\_predict)}**