**WBCD BREAST CANCER AI ASSIGNMENT**

**#import libraries**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

**#Load the data**

df = pd.read\_csv("data.csv")

df.head(7)

**#Count the number of rows and columns in the dataset**

df.shape

**#Count the number of empty (Nan, NAN, na) values in each column**

df.isna().sum()

**#Drop the column with all missing values**

df = df.dropna(axis=1)

**#Get the new count of number of rows and columns**

df.shape

**#Get a count of the number of Malignant (M) or Benign (B) cells**

df['diagnosis'].value\_counts()

**#Visualize the count**

sns.countplot(df['diagnosis'], label= 'count')

**#Look at the data types to see which columns need to be encoded**

df.dtypes

**#Encode the categorical data values**

from sklearn.preprocessing import LabelEncoder

labelEncoder\_Y = LabelEncoder()

df.iloc[: , 1]= labelEncoder\_Y.fit\_transform(df.iloc[:,1].values)

**#Create a pair plot**

sns.pairplot(df.iloc[:,1:5], hue = 'diagnosis')

**#Print the first five rows of new data**

df.head(5)

**#Get the correlation of the columns**

df.iloc[:,1:12].corr()

**#Visualize the correlation**

plt.figure(figsize=(10,10))

sns.heatmap(df.iloc[:,1:12].corr(), annot = True, fmt = '.0%')

**#Split the dataset into independent (x) and dependent (y) data sets**

X = df.iloc[:,2:31].values

Y = df.iloc[:,1].values

**#Split the dataset into 75% training and 25% testing**

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

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.25, random\_state = 0)

**#Scale the data (Feature Scaling)**

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.fit\_transform(X\_test)

X\_train

**#Create a function for the training models**

def models(X\_train, Y\_train):

**#Logistic Regression**

from sklearn.linear\_model import LogisticRegression

**#Create a LR Classifier**

log = LogisticRegression(random\_state = 0)

**#Train the model using the training sets**

log.fit(X\_train, Y\_train)

**#Decision Tree**

from sklearn.tree import DecisionTreeClassifier

**#Create a DT Classifier**

tree = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)

**#Train the model using the training sets**

tree.fit(X\_train, Y\_train)

**#Random Forest Classifier**

from sklearn.ensemble import RandomForestClassifier

**#Create a RF Classifier**

forest = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state=0)

**#Train the model using the training sets**

forest.fit(X\_train, Y\_train)

**#Support Vector Machine Classifier**

from sklearn import svm

**#Create a svm Classifier**

support = svm.SVC(kernel='linear') # Linear Kernel

**#Train the model using the training sets**

support.fit(X\_train, Y\_train)

**#Print the models' accuracy on the training data**

print('[0]Logistic Regression Training Accuracy: ', log.score(X\_train, Y\_train))

print('[1]Decision Tree Training Accuracy: ', tree.score(X\_train, Y\_train))

print('[2]Random Forest Training Accuracy: ', forest.score(X\_train, Y\_train))

print('[3]Support Vector Machine Training Accuracy:', support.score(X\_train, Y\_train))

return log, tree, forest, support

**#Getting all of the models**

model = models(X\_train, Y\_train)

**#Metrics derivation method**

from sklearn.metrics import classification\_report

from sklearn.metrics import accuracy\_score

for i in range(len(model)):

print('Model ', i)

print(classification\_report(Y\_test, model[i].predict(X\_test)))

print(accuracy\_score(Y\_test, model[i].predict(X\_test)))

print()