MACHINE LEARNING INDIVIDUAL ASSIGNMENT-(2) REPORT

ABHIGNA KAPPAGANTULA

SE22UCSE003

CSE-1

IMPORTING ALL THE NECESSARY LIBRARIES:

# importing libraries

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

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

from imblearn.over\_sampling import SMOTE

!pip install imbalanced-learn

from sklearn.metrics import accuracy\_score

LOADING THE DATASET INTO PYTHON:

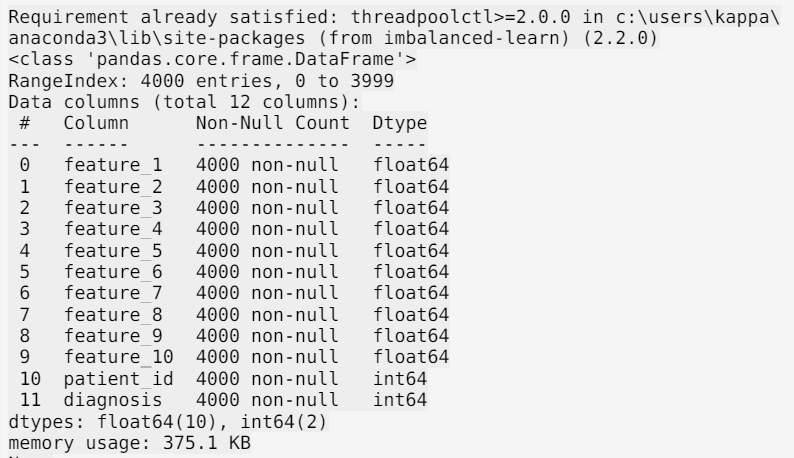
#loading dataset into python

train\_ds=pd.read\_csv("C:\\Users\\kappa\\Downloads\\Disease\_train.csv")

#to identify all features and datatypes

train\_ds.head()

print(train\_ds.info())



CHECKING FOR DUPLICATE ROWS AND REMOVING IF THERE EXISTS ANY:

#to check if there is any duplicate row

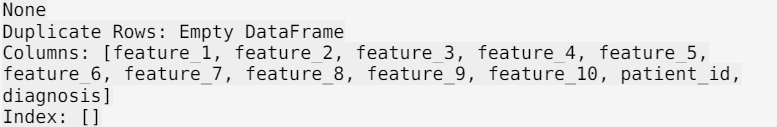
DUPLICATEE\_ROWSS=train\_ds[train\_ds.duplicated()]

#to print duplicate rows

print("Duplicate Rows:",DUPLICATEE\_ROWSS)

#to remove duplicate rows

train\_ds=train\_ds.drop\_duplicates()



MISSING VALUES:

#MISSING VALUE HANDLING

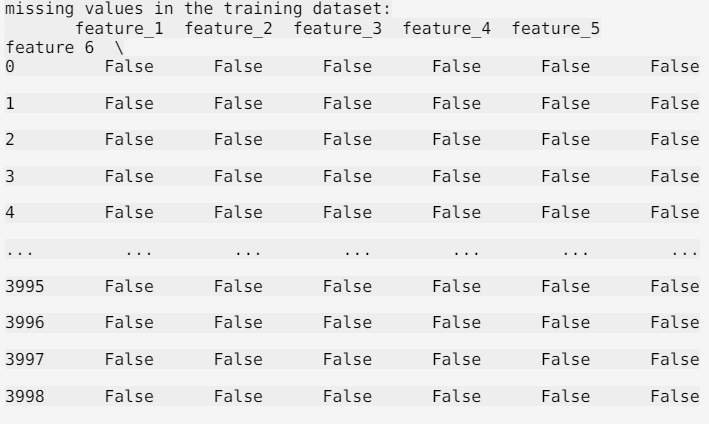
train\_ds.shape

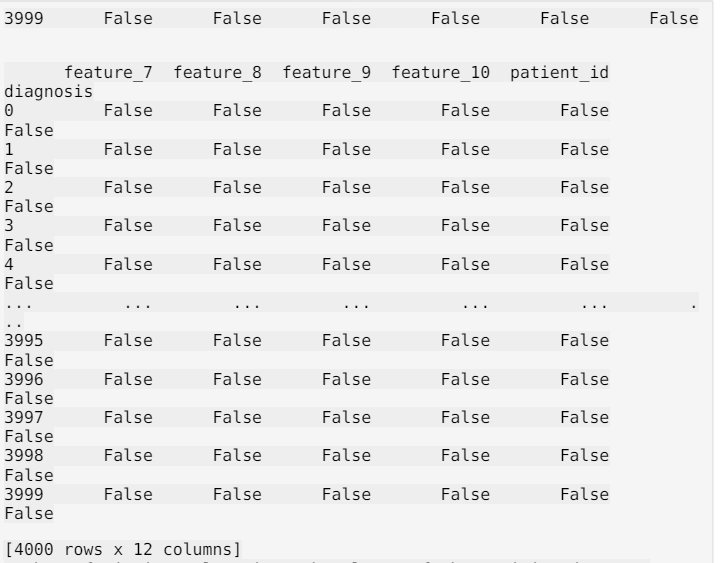
print("missing values in the training dataset:\n",train\_ds.isnull())

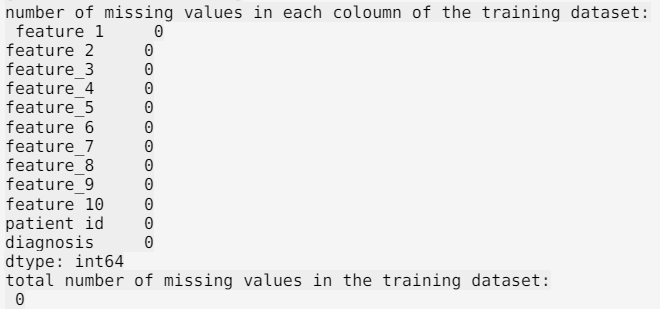
print("number of missing values in each coloumn of the training dataset:\n",train\_ds.isnull().sum())

print("total number of missing values in the training dataset:\n",train\_ds.isnull().sum().sum())

# no null values







SHAPE OF THE TRAINING DATASET:

# number of rows and  columns in the training dataset:

print("SHAPE OF THE TRAINING DATASET:(number of rows, number of columns):",train\_ds.shape)



PREPARING AND ANALYZING THE TRAINING DATA FOR MACHINE LEARNING [ DIAGNOSIS PREDICTION]:

We separate the features from the target variable [ diagnosis ] and we observe and count the unique values and the number of times they occur.

X\_train\_ds = train\_ds.drop(['diagnosis','patient\_id'],axis=1)

Y\_train\_ds = train\_ds['diagnosis']

unique, count = np.unique(Y\_train\_ds, return\_counts = True)

Y\_train\_ds\_value\_count = { k:v for (k,v) in zip(unique,count)}

print("SET OF ALL UNIQUE VALUES AND THE NUMBER OF TIMES THEY OCURRED:\n {(unique values:count of each unique value)}:", Y\_train\_ds\_value\_count)



IMPLEMENTING SMOTE FOR BALANCING AND TRAINING A RANDOM FOREST CLASSIFIER:

This code snippet first balances the training dataset using SMOTE (Synthetic Minority Over-sampling Technique) to ensure equal representation of classes. It then splits the balanced data into training and validation sets. A Random Forest Classifier is trained on the new training set, and its performance is evaluated using accuracy on the validation set. The code prints the distribution of the classes in the balanced dataset and the accuracy of the classifier.

SM = SMOTE(random\_state=12,sampling\_strategy=1.0)

X\_train\_new, Y\_train\_new = SM.fit\_resample(X\_train\_ds,Y\_train\_ds)

unique, count = np.unique(Y\_train\_new, return\_counts = True)

Y\_train\_smotevaluecount = { k:v for (k,v) in zip(unique,count)}

print(Y\_train\_smotevaluecount)

X\_train\_new, X\_val, Y\_train\_new, Y\_val = train\_test\_split(X\_train\_new,Y\_train\_new,test\_size=0.2)

RF = RandomForestClassifier(n\_estimators=50,max\_depth=15)

RF.fit(X\_train\_new,Y\_train\_new)

Y\_pred = RF.predict(X\_val)

Accuracy = accuracy\_score(Y\_val,Y\_pred)

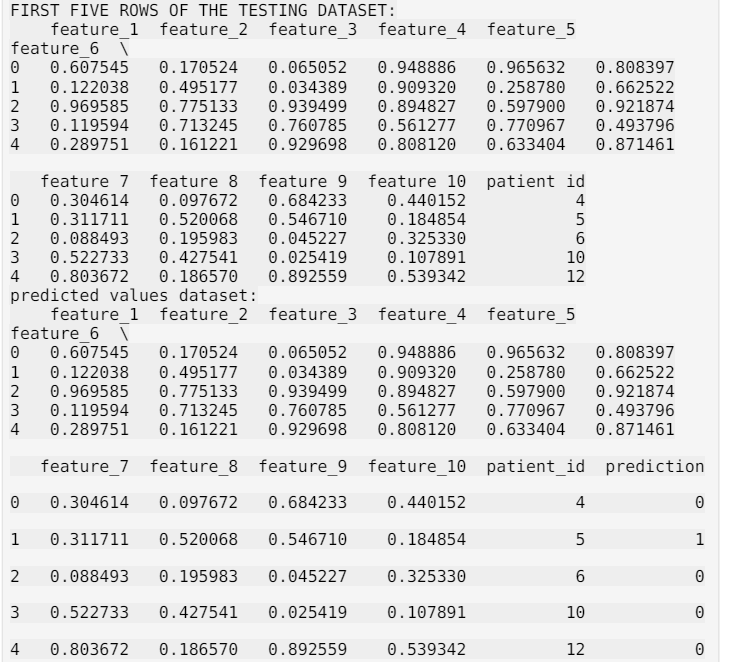
print("Accuracy Score : ", Accuracy)



TESTING DATASET:

test\_dataset = pd.read\_csv("C:\\Users\\kappa\\Downloads\\Disease\_test.csv")

print("FIRST FIVE ROWS OF THE TESTING DATASET: \n",test\_dataset.head())



PREPARING AND MODIFYING TEST DATASET FOR PREDICTIONS:

The modified test dataset, now including the prediction column, is saved to a new CSV file named 'se22ucse003\_predictions.csv'. We remove several feature columns from the DataFrame, leaving only the relevant columns for the final output, and prints the first few rows of this modified DataFrame.

# Drop the 'patient\_id' column to prepare the test dataset for predictions

X\_test\_dataset = test\_dataset.drop(['patient\_id'],axis=1)

# Make predictions using the RandomForest model

prediction\_test\_ds = RF.predict(X\_test\_dataset)

# Add the predictions to the test DataFrame

test\_dataset['prediction'] = prediction\_test\_ds

# Save the modified test DataFrame to a new CSV file

test\_dataset.to\_csv('se22ucse003\_predictions.csv', index=False)

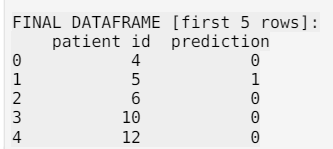
# Load the final CSV file to ensure it was saved correctly

final = pd.read\_csv('se22ucse003\_predictions.csv')

print("predicted values dataset:\n",final.head())

final.drop(['feature\_1','feature\_2','feature\_3','feature\_4','feature\_5','feature\_6','feature\_7','feature\_8','feature\_9','feature\_10'],axis=1,inplace=True)

print("FINAL DATAFRAME [first 5 rows]:\n",final.head())



final.to\_csv('se22ucse003\_predictions.csv',index=False)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*