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())
```

```
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\kappa\
anaconda3\lib\site-packages (from imbalanced-learn) (2.2.0)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   feature_1   4000 non-null   float64
 1   feature_2   4000 non-null   float64
 2   feature_3   4000 non-null   float64
 3   feature_4   4000 non-null   float64
 4   feature_5   4000 non-null   float64
 5   feature_6   4000 non-null   float64
 6   feature_7   4000 non-null   float64
 7   feature_8   4000 non-null   float64
 8   feature_9   4000 non-null   float64
 9   feature_10  4000 non-null   float64
 10  patient_id  4000 non-null   int64
 11  diagnosis   4000 non-null   int64
dtypes: float64(10), int64(2)
memory usage: 375.1 KB
```

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()
```

```
None
Duplicate Rows: Empty DataFrame
Columns: [feature_1, feature_2, feature_3, feature_4, feature_5,
feature_6, feature_7, feature_8, feature_9, feature_10, patient_id,
diagnosis]
Index: []
```

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
```

```
missing values in the training dataset:
       feature_1  feature_2  feature_3  feature_4  feature_5
feature 6  \
0          False      False      False      False      False      False

1          False      False      False      False      False      False

2          False      False      False      False      False      False

3          False      False      False      False      False      False

4          False      False      False      False      False      False

...          ...        ...        ...        ...        ...        ...

3995       False      False      False      False      False      False

3996       False      False      False      False      False      False

3997       False      False      False      False      False      False

3998       False      False      False      False      False      False
```

```
3999       False      False      False      False      False      False


      feature_7  feature_8  feature_9  feature_10  patient_id
diagnosis
0          False      False      False       False       False
False
1          False      False      False       False       False
False
2          False      False      False       False       False
False
3          False      False      False       False       False
False
4          False      False      False       False       False
False
...          ...        ...        ...         ...         ...         .
..
3995       False      False      False       False       False
False
3996       False      False      False       False       False
False
3997       False      False      False       False       False
False
3998       False      False      False       False       False
False
3999       False      False      False       False       False
False

[4000 rows x 12 columns]
```

```
number of missing values in each coloumn of the training dataset:
 feature 1     0
feature 2     0
feature_3     0
feature_4     0
feature_5     0
feature 6     0
feature_7     0
feature_8     0
feature_9     0
feature 10    0
patient id    0
diagnosis     0
dtype: int64
total number of missing values in the training dataset:
 0
```

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)
```

```
SHAPE OF THE TRAINING DATASET:(number of rows, number of columns):
(4000, 12)
```

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)
```

```
SET OF ALL UNIQUE VALUES AND THE NUMBER OF TIMES THEY OCURRED:
 {(unique values:count of each unique value)}: {0: 3804, 1: 196}
{0: 3804, 1: 3804}
```

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)
```

```
Accuracy Score :  0.8843626806833115
```

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())
```

```
FIRST FIVE ROWS OF THE TESTING DATASET:
    feature_1  feature_2  feature_3  feature_4  feature_5
feature_6  \
0   0.607545   0.170524   0.065052   0.948886   0.965632   0.808397
1   0.122038   0.495177   0.034389   0.909320   0.258780   0.662522
2   0.969585   0.775133   0.939499   0.894827   0.597900   0.921874
3   0.119594   0.713245   0.760785   0.561277   0.770967   0.493796
4   0.289751   0.161221   0.929698   0.808120   0.633404   0.871461

   feature_7  feature_8  feature_9  feature_10  patient_id
0   0.304614   0.097672   0.684233    0.440152           4
1   0.311711   0.520068   0.546710    0.184854           5
2   0.088493   0.195983   0.045227    0.325330           6
3   0.522733   0.427541   0.025419    0.107891          10
4   0.803672   0.186570   0.892559    0.539342          12
predicted values dataset:
    feature_1  feature_2  feature_3  feature_4  feature_5
feature_6  \
0   0.607545   0.170524   0.065052   0.948886   0.965632   0.808397
1   0.122038   0.495177   0.034389   0.909320   0.258780   0.662522
2   0.969585   0.775133   0.939499   0.894827   0.597900   0.921874
3   0.119594   0.713245   0.760785   0.561277   0.770967   0.493796
4   0.289751   0.161221   0.929698   0.808120   0.633404   0.871461

   feature_7  feature_8  feature_9  feature_10  patient_id  prediction

0   0.304614   0.097672   0.684233    0.440152           4           0

1   0.311711   0.520068   0.546710    0.184854           5           1

2   0.088493   0.195983   0.045227    0.325330           6           0

3   0.522733   0.427541   0.025419    0.107891          10           0

4   0.803672   0.186570   0.892559    0.539342          12           0
```

PREPARING AND MODIFYING TEST DATASET FOR PREDICTIONS:

The modified test dataset, now including the prediction column, is saved to a new CSV file named `'se22ucse_003_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','featu
re_6','feature_7','feature_8','feature_9','feature_10'],axis=1,inplace=True)
print("FINAL DATAFRAME [first 5 rows]:\n",final.head())
```

```
FINAL DATAFRAME [first 5 rows]:
    patient id  prediction
0            4           0
1            5           1
2            6           0
3           10           0
4           12           0
```

```
final.to_csv('se22ucse003_predictions.csv',index=False)
```

****