Experiment 10 - Implementation of supervised learning algorithm Random Forest

Roll No.	37
Name	Mikil Lalwani
Class	D20 B
Subject	Data Science Lab
LO Mapped	L4: Develop real life applications using learning concepts. L5: Evaluate performance of applications.

Aim: To implement supervised learning algorithm Random Forest.

Random forest:

Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

Decision trees

Since the random forest model is made up of multiple decision trees, it would be helpful to start by describing the decision tree algorithm briefly. Decision trees start with a basic question, such as, “Should I surf?” From there, you can ask a series of questions to determine an answer, such as, “Is it a long period swell?” or “Is the wind blowing offshore?”. These questions make up the decision nodes in the tree, acting as a means to split the data. Each question helps an individual to arrive at a final decision, which would be denoted by the leaf node. Observations that fit the criteria will follow the “Yes” branch and those that don’t will follow the alternate path. Decision trees seek to find the best split to subset the data, and they are typically trained through the Classification and Regression Tree (CART) algorithm. Metrics, such as Gini impurity, information gain, or mean square error (MSE), can be used to evaluate the quality of the split.

Ensemble methods

Ensemble learning methods are made up of a set of classifiers—e.g. decision trees—and their predictions are aggregated to identify the most popular result. The most well-known ensemble methods are bagging, also known as bootstrap aggregation, and boosting. In 1996, Leo Breiman (link resides outside ibm.com) (PDF, 810 KB) introduced the bagging method; in this method, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once. After several data samples are generated, these models are then trained independently, and depending on the type of task—i.e. regression or classification—the average or majority of those predictions yield a more accurate estimate. This approach is commonly used to reduce variance within a noisy dataset.

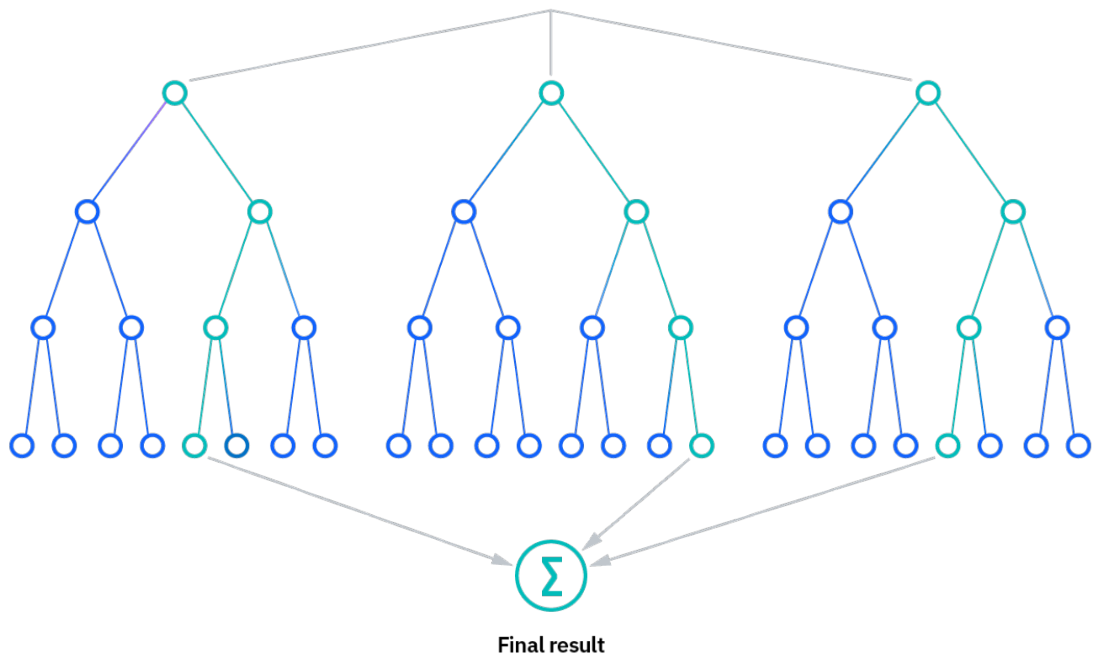
Random forest algorithm

The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees. Feature randomness, also known as feature bagging or “the random subspace method”(link resides outside ibm.com) (PDF, 121 KB), generates a random subset of features, which ensures low correlation among decision trees. This is a key difference between decision trees and random forests. While decision trees consider all the possible feature splits, random forests only select a subset of those features.

How it works

Random forest algorithms have three main hyperparameters, which need to be set before training. These include node size, the number of trees, and the number of features sampled. From there, the random forest classifier can be used to solve for regression or classification problems.

The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample. Of that training sample, one-third of it is set aside as test data, known as the out-of-bag (oob) sample, which we'll come back to later. Another instance of randomness is then injected through feature bagging, adding more diversity to the dataset and reducing the correlation among decision trees. Depending on the type of problem, the determination of the prediction will vary. For a regression task, the individual decision trees will be averaged, and for a classification task, a majority vote—i.e. the most frequent categorical variable—will yield the predicted class. Finally, the oob sample is then used for cross-validation, finalizing that prediction.



Python Library Function Used:

Topic: Lung Cancer Detection.
Dataset: https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer

1. **pandas -**
   Pandas is an open-source library in Python that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library of Python. Pandas is fast and it has high performance & productivity for users.

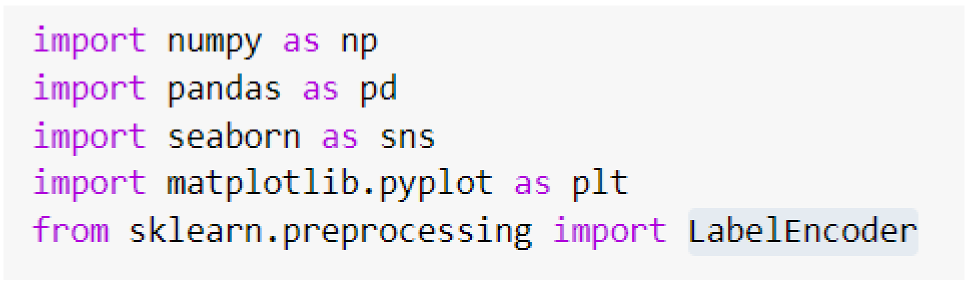
2. **Seaborn -**
   Seaborn is a Python data visualisation library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

3. **LabelEncoder -**
   LabelEncoder is a utility class from the scikit-learn (sklearn) library in Python used for encoding categorical labels into numerical values. It assigns a unique integer to each distinct label in a categorical variable, effectively converting them into a format that machine learning algorithms can work with. This transformation is particularly useful when dealing with classification tasks where the target variable or class labels need to be represented numerically. LabelEncoder is a simple and commonly used tool for this purpose in scikit-learn.

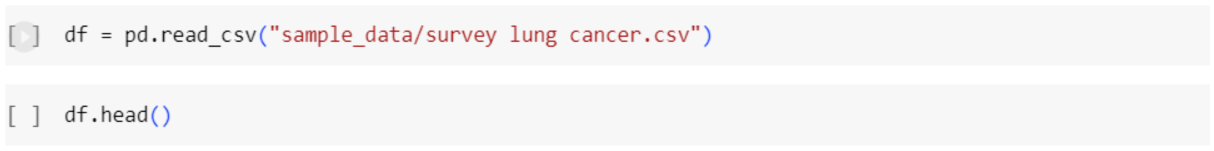
4. **RandomForestClassifier -**
   RandomForestClassifier is a machine learning algorithm implemented in scikit-learn (sklearn) that belongs to the ensemble learning category. It is an ensemble of decision trees that are trained on different subsets of the data and combined to make predictions. RandomForestClassifier is used primarily for classification tasks. It works by constructing multiple decision trees during training and then averaging or taking a majority vote of their predictions to make more accurate and robust classification decisions. This algorithm is known for its high predictive accuracy and ability to handle complex datasets, making it a popular choice for a wide range of classification problems in machine learning.

Code and Observation:

1. Importing dataset

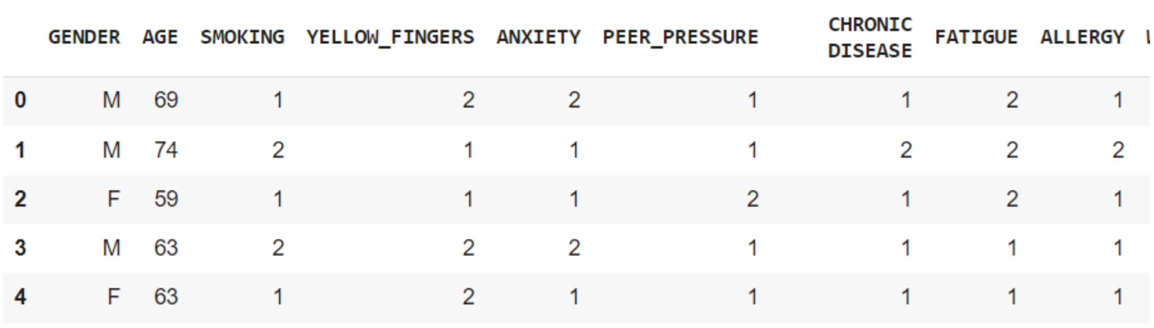


```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
```



```
[ ] df = pd.read_csv("sample_data/survey lung cancer.csv")
```

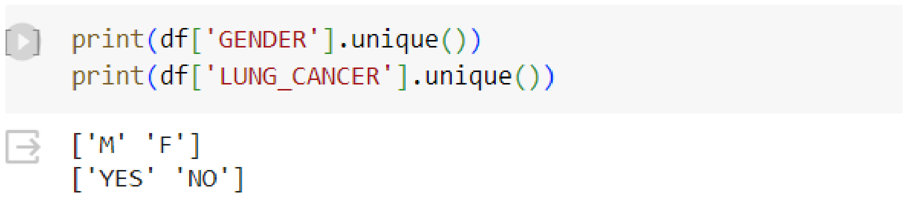
```
[ ] df.head()
```



	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY
0	M	69	1	2	2	1	1	2	1
1	M	74	2	1	1	1	2	2	2
2	F	59	1	1	1	2	1	2	1
3	M	63	2	2	2	1	1	1	1
4	F	63	1	2	1	1	1	1	1

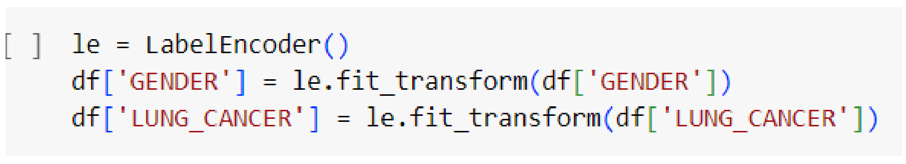
2. Cleaning dataset

3. Label encoding string values into numerical values.



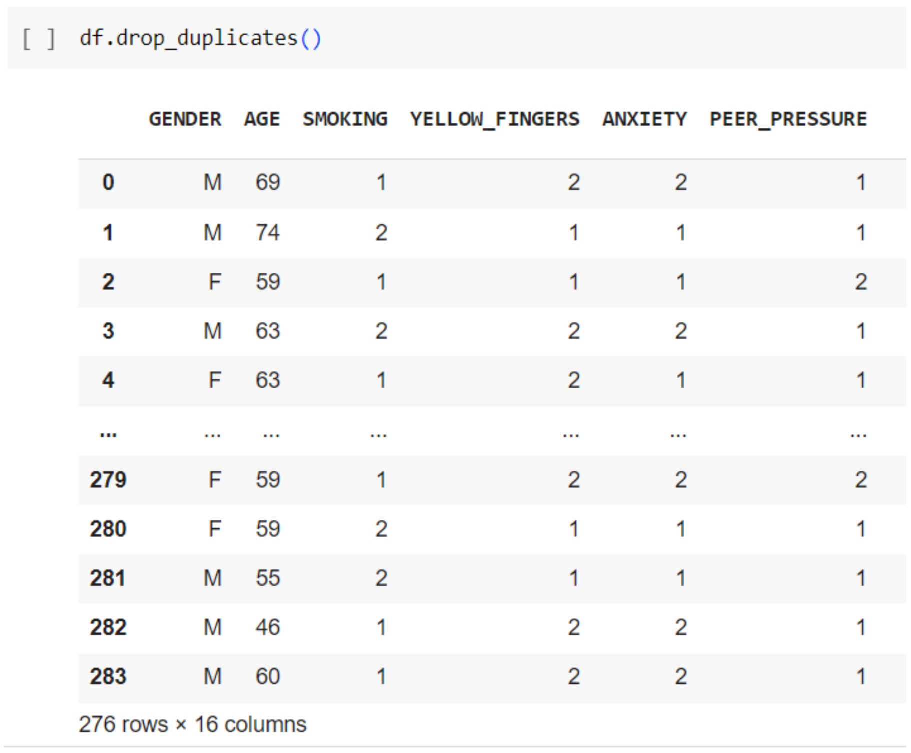
```
print(df['GENDER'].unique())
print(df['LUNG_CANCER'].unique())
```

```
['M' 'F']
['YES' 'NO']
```

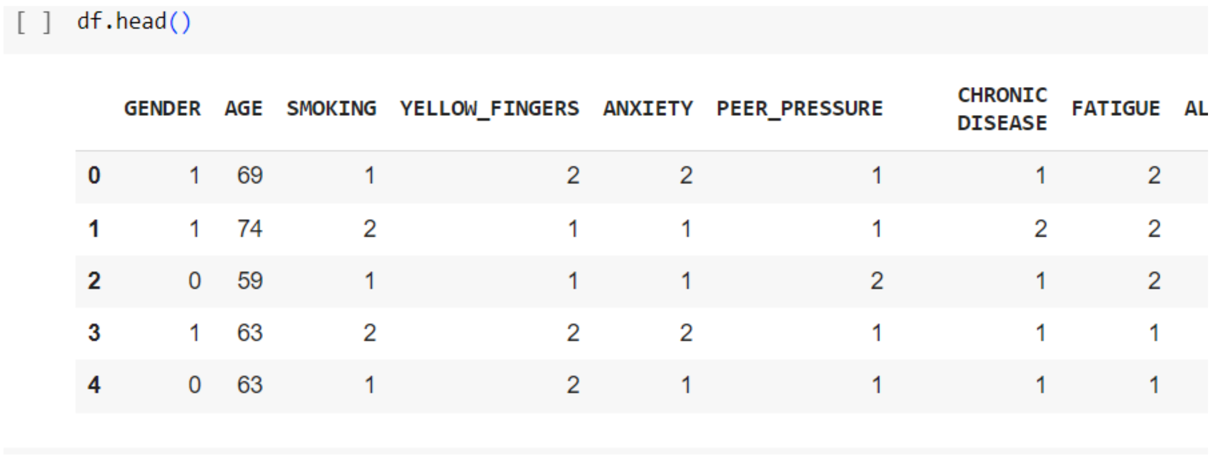


```
[ ] le = LabelEncoder()
    df['GENDER'] = le.fit_transform(df['GENDER'])
    df['LUNG_CANCER'] = le.fit_transform(df['LUNG_CANCER'])
```

Drop Duplicates Values



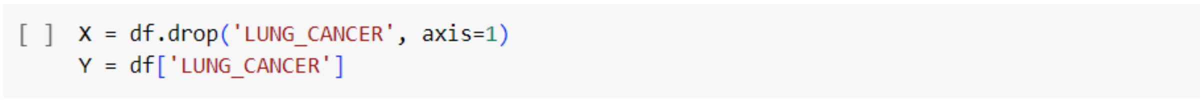
```
[ ]  df.drop_duplicates()

        GENDER  AGE  SMOKING  YELLOW_FINGERS  ANXIETY  PEER_PRESSURE
   0        M    69        1               2        2              1
   1        M    74        2               1        1              1
   2        F    59        1               1        1              2
   3        M    63        2               2        2              1
   4        F    63        1               2        1              1
  ...     ...   ...      ...             ...      ...            ...
 279        F    59        1               2        2              2
 280        F    59        2               1        1              1
 281        M    55        2               1        1              1
 282        M    46        1               2        2              1
 283        M    60        1               2        2              1
276 rows × 16 columns
```



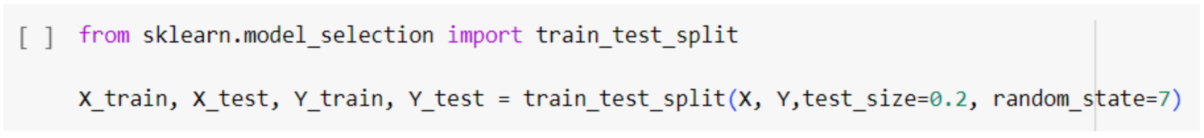
```
[ ]  df.head()

                                                              CHRONIC
     GENDER  AGE  SMOKING  YELLOW_FINGERS  ANXIETY  PEER_PRESSURE  DISEASE  FATIGUE  AL
  0       1   69        1               2        2              1        1        2
  1       1   74        2               1        1              1        2        2
  2       0   59        1               1        1              2        1        2
  3       1   63        2               2        2              1        1        1
  4       0   63        1               2        1              1        1        1
```

GENDER: 1 - Male, 0 - Female LUNG_CANCER: 1 - YES, 0 - NO

4. Splitting dataset into train and test set.

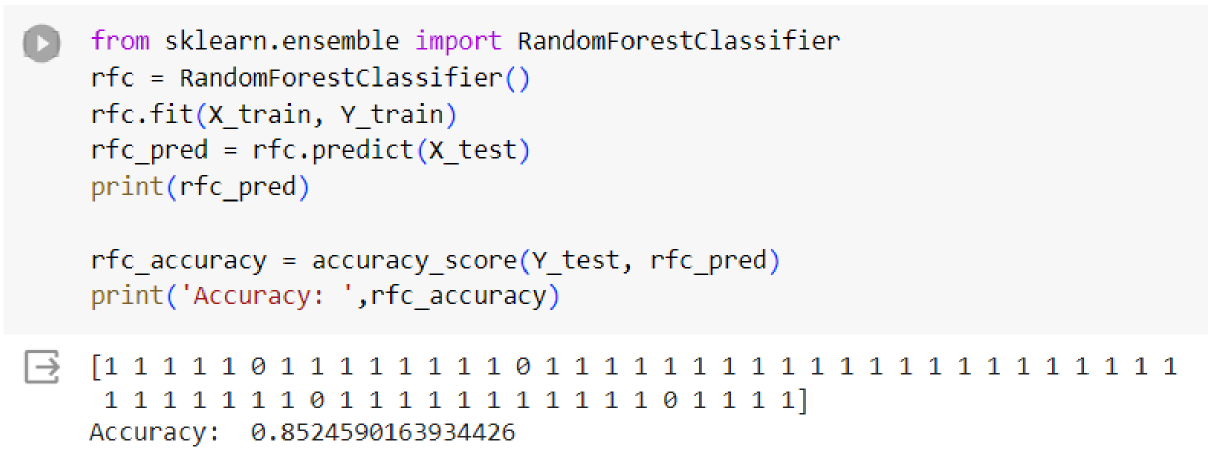


```
[ ]  X = df.drop('LUNG_CANCER', axis=1)
     Y = df['LUNG_CANCER']
```



```
[ ]  from sklearn.model_selection import train_test_split

     X_train, X_test, Y_train, Y_test = train_test_split(X, Y,test_size=0.2, random_state=7)
```

5. Creating Random Forest classification model.



```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, Y_train)
rfc_pred = rfc.predict(X_test)
print(rfc_pred)

rfc_accuracy = accuracy_score(Y_test, rfc_pred)
print('Accuracy: ',rfc_accuracy)
```

```
[1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1]
Accuracy:  0.8524590163934426
```

Conclusion:

Understood and implemented Random Forest classifier on lung cancer detection dataset.