Experiment 9 - Implementation of supervised learning algorithm Ada-Boosting

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 Ada-Boost.

Theory:

What is AdaBoost?
AdaBoost short for Adaptive Boosting is an ensemble learning used in machine learning for classification and regression problems. The main idea behind AdaBoost is to iteratively train the weak classifier on the training dataset with each successive classifier giving more weightage to the data points that are misclassified. The final AdaBoost model is decided by combining all the weak classifier that has been used for training with the weightage given to the models according to their accuracies. The weak model which has the highest accuracy is given the highest weightage while the model which has the lowest accuracy is given a lower weightage.

Institution Behind AdaBoost Algorithm

AdaBoost techniques combine many weak machine-learning models to create a powerful classification model for the output. The steps to build and combine these models are as

Step1 – Initialize the weights

- For a dataset with N training data points instances, initialize **N** W_{i} weights for each data point with W_{i} = \frac{1}{N}

Step2 – Train weak classifiers

- Train a weak classifier M_k where k is the current iteration
- The weak classifier we are training should have an accuracy greater than 0.5 which means it should be performing better than a naive guess

Step3 – Calculate the error rate and importance of each weak model M_k

- Calculate rate *error_rate* for every weak classifier M_k on the training dataset
- Calculate the importance of each model α_k using formula \alpha_k = \frac{1}{2} \ln{\frac{1 – error_k}{error_k}}

Step4 – Update data point weight for each data point W_i

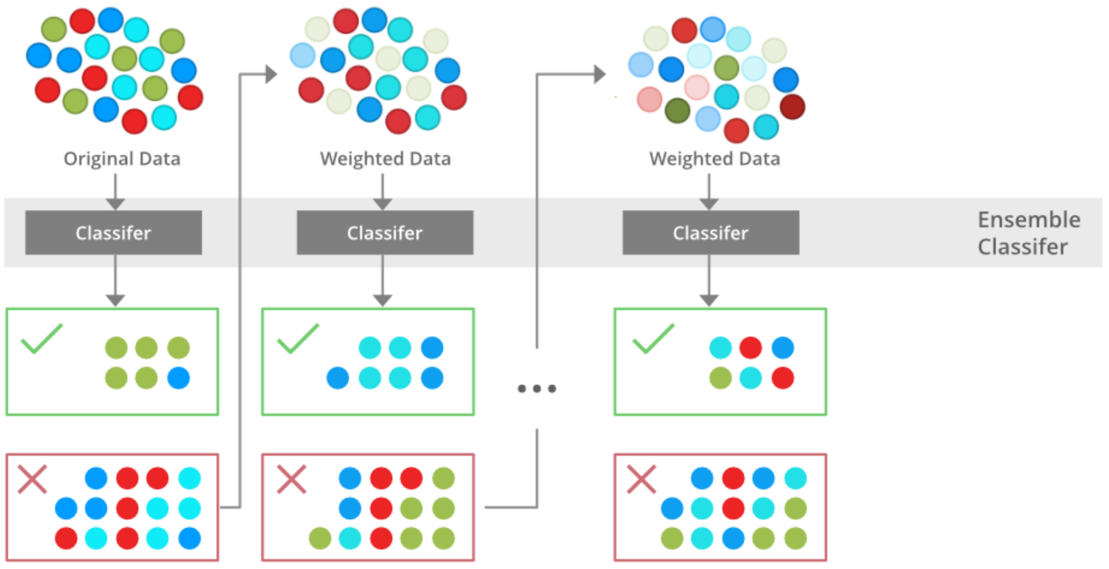
- After applying the weak classifier model to the training data we will update the weight assigned to the points using the accuracy of the model. The formula for updating the weights will be w_i = w_i \exp{(-\alpha_k y_i M_k(x_i))} . Here y_i is the true output and X_i is the corresponding input vector

Step5 – Normalize the Instance weight

- We will normalize the instance weight so that they can be summed up to 1 using the formula W_i = W_i / sum(W)

Step6 – Repeat steps 2-5 for K iterations

- We will train K classifiers and will calculate model importance and update the instance weights using the above formula
- The final model **M(X)** will be an ensemble model which is obtained by combining these weak models weighted by their model weights



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. **AdaBoostClassifier -**
   The AdaBoostClassifier used in the provided code is a machine learning algorithm from scikit-learn that belongs to the ensemble learning family. Specifically, it stands for Adaptive Boosting. AdaBoost works by combining multiple weak learners (typically decision trees with limited depth) to create a strong and accurate classifier. It assigns different weights to the training instances and adjusts them in subsequent iterations to focus on the samples that are difficult to classify correctly. By doing so, AdaBoost builds a strong classifier by sequentially emphasizing the weaknesses of the previous models, ultimately resulting in improved overall predictive performance. It is commonly used for classification tasks and is known for its ability to handle complex datasets and improve classification accuracy.

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

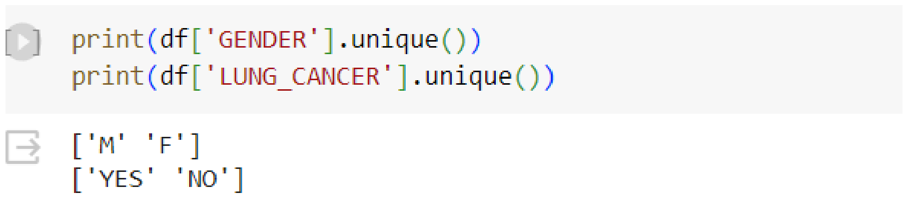
2. Cleaning dataset

Drop Duplicates Values

```
[ ] df.drop_duplicates()
```

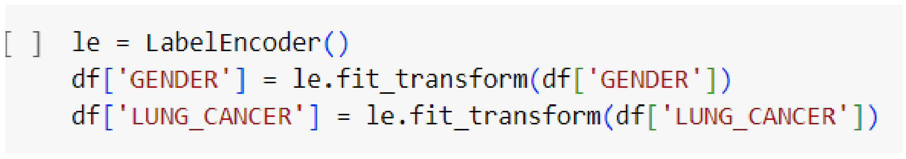
	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE
0	M	69	1	2	2	1
1	M	74	2	1	1	1
2	F	59	1	1	1	2
3	M	63	2	2	2	1
4	F	63	1	2	1	1
...	...	...	...	...	...	...
279	F	59	1	2	2	2
280	F	59	2	1	1	1
281	M	55	2	1	1	1
282	M	46	1	2	2	1
283	M	60	1	2	2	1

3. Label encoding string values into numerical values.

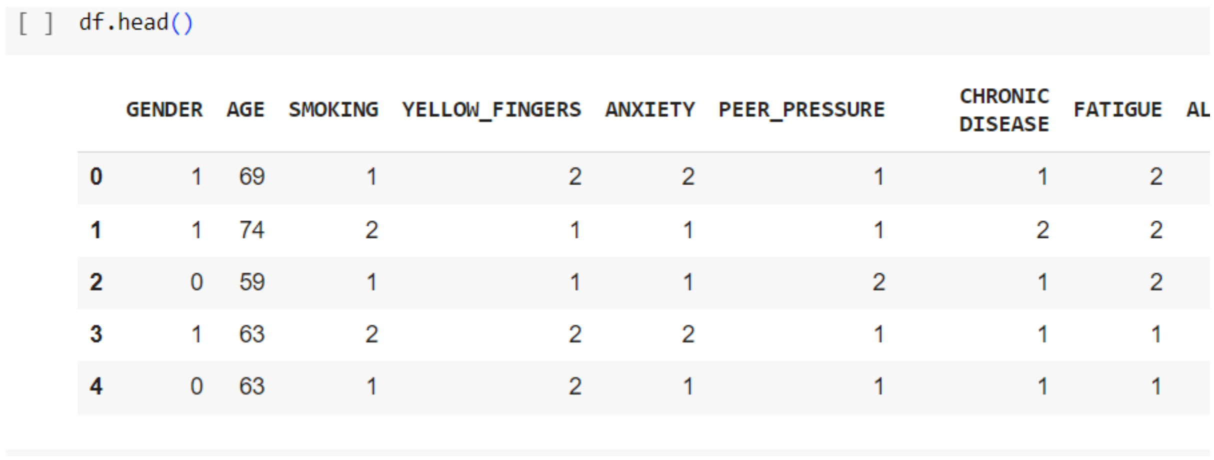


```
print(df['GENDER'].unique())
print(df['LUNG_CANCER'].unique())
```

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



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

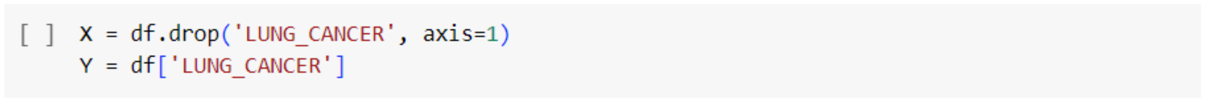


```
[ ] df.head()
```

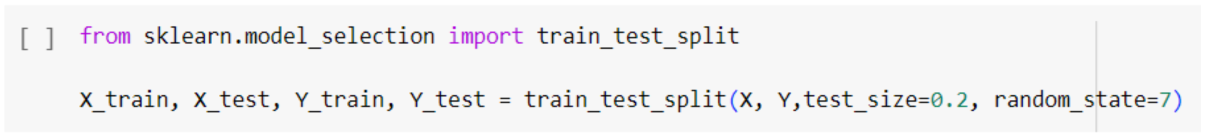
	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	AL
0	1	69	1	2	2	1	1	2	
1	1	74	2	1	1	1	2	2	
2	0	59	1	1	1	2	1	2	
3	1	63	2	2	2	1	1	1	
4	0	63	1	2	1	1	1	1	

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

4. Splitting dataset into train and test set.



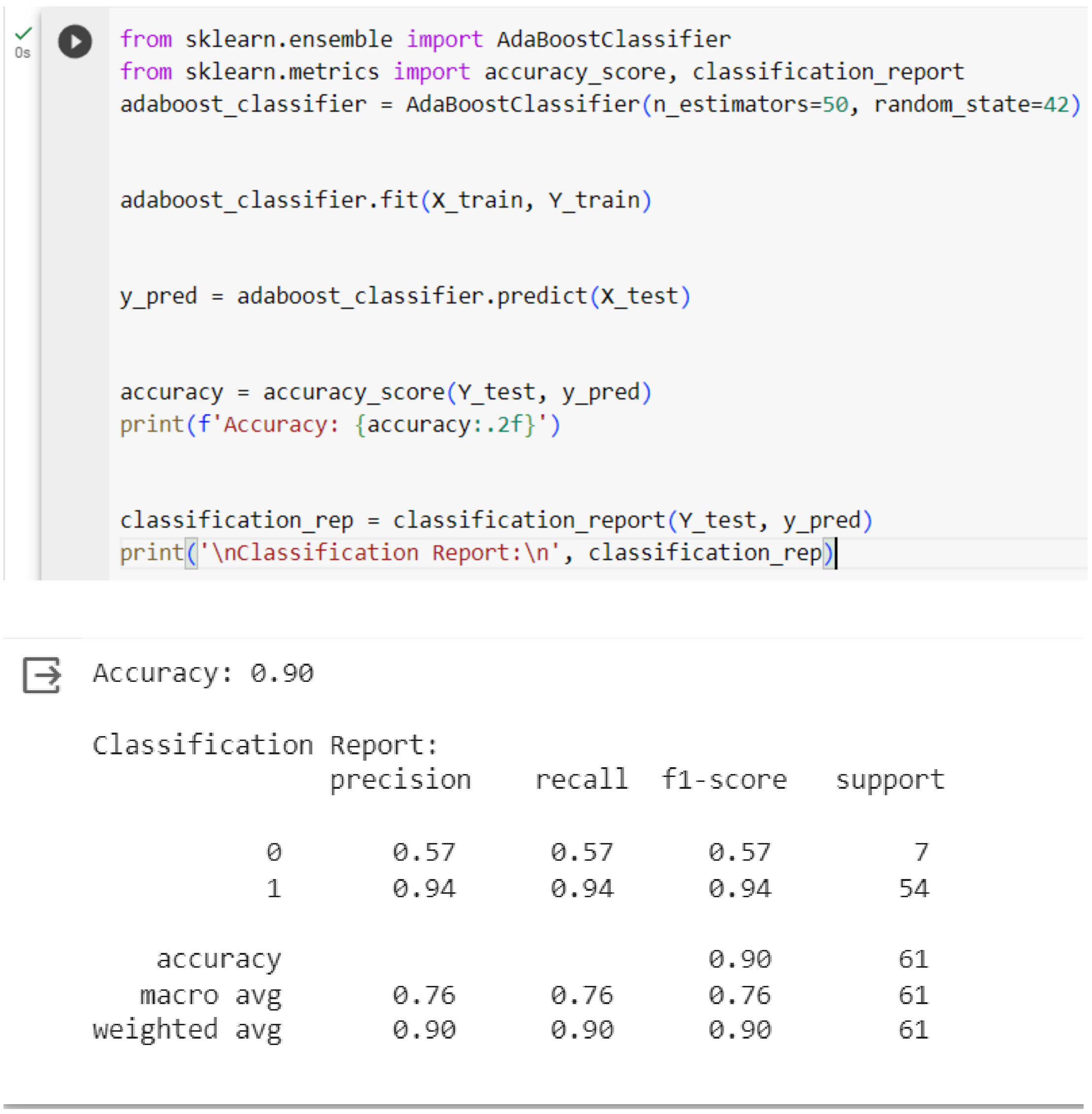
```
[ ] X = df.drop('LUNG_CANCER', axis=1)
    Y = df['LUNG_CANCER']
```



```
[ ] 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 an AdaBoost classifier model.



```python
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, classification_report
adaboost_classifier = AdaBoostClassifier(n_estimators=50, random_state=42)


adaboost_classifier.fit(X_train, Y_train)


y_pred = adaboost_classifier.predict(X_test)


accuracy = accuracy_score(Y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')


classification_rep = classification_report(Y_test, y_pred)
print('\nClassification Report:\n', classification_rep)
```

```
Accuracy: 0.90

Classification Report:
               precision    recall  f1-score   support

           0       0.57      0.57      0.57         7
           1       0.94      0.94      0.94        54

    accuracy                           0.90        61
   macro avg       0.76      0.76      0.76        61
weighted avg       0.90      0.90      0.90        61
```

Conclusion:

Understood and implemented AdaBoost classifier on lung cancer detection dataset.