Name: Arlene Antony D'costa
Seating Pin: 79
SUID: 594303899

CIS 662 HW8 Report: Adaboost classifier

Dataset selected is **71-80.csv**

Data has also been split in the same way as the previous homework 80:20 to compare the results.

```
Calculate ratio between cit_2021 and cit_2022

data["ratio_21_22"] = round(data["cit_2022"] / data["cit_2021"],2)
data.fillna(0, inplace=True)
✓ 0.0s

data["category"] = pd.cut(data["ratio_21_22"],bins=[-np.inf,1.05, 1.16, np.inf],labels=["Low", "Medium", "High"])
✓ 0.0s
```

Steps to convert continuous target variable to categorical target variable:

Step 1: Data contains all continuous numerical values, however, to convert the target variable to a categorical variable we first take the ratio of **cit_2022/cit_2021** and round it to **2 decimals.**

Step 2: Binning strategy is used to distribute the data present in different bins i.e Separate data into different categories.
Thus 3, separate labelled categories can be obtained as follows:
1. Low (<1.05): Values between -np.inf (inclusive) and 1.05 (exclusive)
2. Medium (1.05-1.15): Values between 1.05 (inclusive) and 1.16 (exclusive)
3. High (>1.15): Values greater than or equal to 1.16 will be labelled as "High".

I have **not scaled/normalised** the data as scaling/normalisation has no significant influence on the output category for Decision Trees/ Random Forest/ Boosting.

We can observe from the above logic that the **ratio of cit_2022/cit_2021** influences the citation category for the citation value to be predicted.

Boosting is an ensemble method where models are trained using decision stumps (decision trees with smaller depth) i.e weak learner as they are faster to train and resistant to overfitting.

In boosting, training is not **done in parallel like RF** and misclassified data points are considered for the decision stump with minimum error (maximum no of correctly classified data points). Misclassified data points are assigned **higher weights** and have higher chance of consideration for the next boosting round with the **condition satisfying the misclassified examples**.

Name: Arlene Antony D'costa
Seating Pin: 79
SUID: 594303899

Step 1: Assign equal weights to data points (1/ (no of data points))
Step 2: Calculate the error rate e_t [Probability of misclassified examples]
Step 3: Calculate the importance factor which is given by:

$$\alpha_t = \eta \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

Where α_t ranges from -inf if e_t is 1 and +inf if e_t is 0
Step 4: Calculate new weights based on the below methods:

$$w_i^{j+1} = \frac{w_i^j}{z_j} * \begin{cases} exp^{-\alpha_i} \; if \; C_i(x_j) = y_j \\ exp^{\alpha_i} \; if \; C_i(x_j) \neq y_j \end{cases}$$

Where z_j is the normalisation factor for the weights to remain distributed in the next round. If Ci(xj) != y_j then the multiplication becomes negative and the α_i becomes positive so the **weight becomes large** else the **weight becomes small**.

I have provided 3 hyperparameter value to the Adaboost algorithm:

1) Learning rate
2) Stump Depth
3) No of samples

```
params = {
    'base_estimator__max_depth': np.arange(1, 11),
    'n_estimators': [10, 50, 100, 200, 300, 500],
    'learning_rate': np.arange(0.1, 1.0, 0.1),
}
✓ 0.0s
```

I am using **RandomizedSearchCV** for evaluating the hyperparameters for AdaboostClassifier model.

Train the Adaboost Model

```
base_estimator = DecisionTreeClassifier(random_state=20)
search = RandomizedSearchCV(AdaBoostClassifier(base_estimator, random_state=20), params,
                            n_iter=50, cv=3, n_jobs=-1)
search.fit(X_train, y_train)
✓ 13.7s
```

```
c:\Users\Arlene\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\model_selectio
  self.best_estimator_ = clone(base_estimator).set_params(
```

▸ RandomizedSearchCV
▸ estimator: AdaBoostClassifier
▸ estimator: DecisionTreeClassifier
▸ DecisionTreeClassifier

Name: Arlene Antony D'costa
Seating Pin: 79
SUID: 594303899

OUTPUT:

Output Classification matrix for Adaboost:

```
    # Print the classification report
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
 ✓  0.0s
```

```
Classification Report:
              precision    recall  f1-score   support

        High       1.00      1.00      1.00         3
         Low       0.92      0.86      0.89        14
      Medium       0.50      0.67      0.57         3

    accuracy                           0.85        20
   macro avg       0.81      0.84      0.82        20
weighted avg       0.87      0.85      0.86        20
```

Accuracy for Adaboost:

```
    y_pred = best_clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Test Accuracy with Best Parameters: {accuracy:.4f}")
 ✓  0.0s
```

```
Test Accuracy with Best Parameters: 0.8500
```

F1-score for Adaboost:

```
    from sklearn.metrics import f1_score
    f1 = f1_score(y_test, y_pred, average='weighted')

    print(f"F1-Score: {f1:.2f}")
 ✓  0.0s
```

```
F1-Score: 0.86
```

Name: Arlene Antony D'costa
Seating Pin: 79
SUID: 594303899

Comparison of F1-score and Accuracy of all the models:

	Logistic Regression	Neural Network	Random Forest	Adaboost
Accuracy	**0.55**	**0.65**	**0.75**	**0.85**
F1-score	**0.55**	**0.67**	**0.75**	**0.86**

CONCLUSION:

F1score for classification with Adaboost has yielded the best results of **86%** as compared to all the other ML algorithms with the same dataset 71-80.csv.
Accuracy obtained is 85% with Adaboost model.

Base classifier for Adaboost was Decision trees.

There are certain reasons why Adaboost might outperform other ML models:

a) Capable of capturing **complex non-linearity in data** and are **robust against overfitting** because they use many weak learners that underfit (high bias) and combine these predictions into a strong learner. Hence, Adaboost is better than Logistic Regression.
b) Random Forests use only subset of features as they want to capture **the variance in the data** by preventing repetition of trees by always using highly correlated features and it can be observed in the previous homework that it provided maximum feature importance to cit_2017 column as it showed mean decrease in Gini index. Since the RF uses plurality or majority voting to determine the results, cit_2017 valued classifier trees were used to capture incorrect results.
c) Decision trees with Adaboost might work better than Neural Networks because it draws vertical lines/ horizontal lines between different classes to minimize entropy capturing intricate patterns especially when the classes are linearly separable. Neural Networks capture patterns depending on the hidden neurons and activation functions when inputs fall into certain feature spaces. However, trees are deterministic hence they work better with tabular data and rule-based system and might perform better in my case where certain features directly impact the final variable.

Code for identifying the most important feature selected for classification.

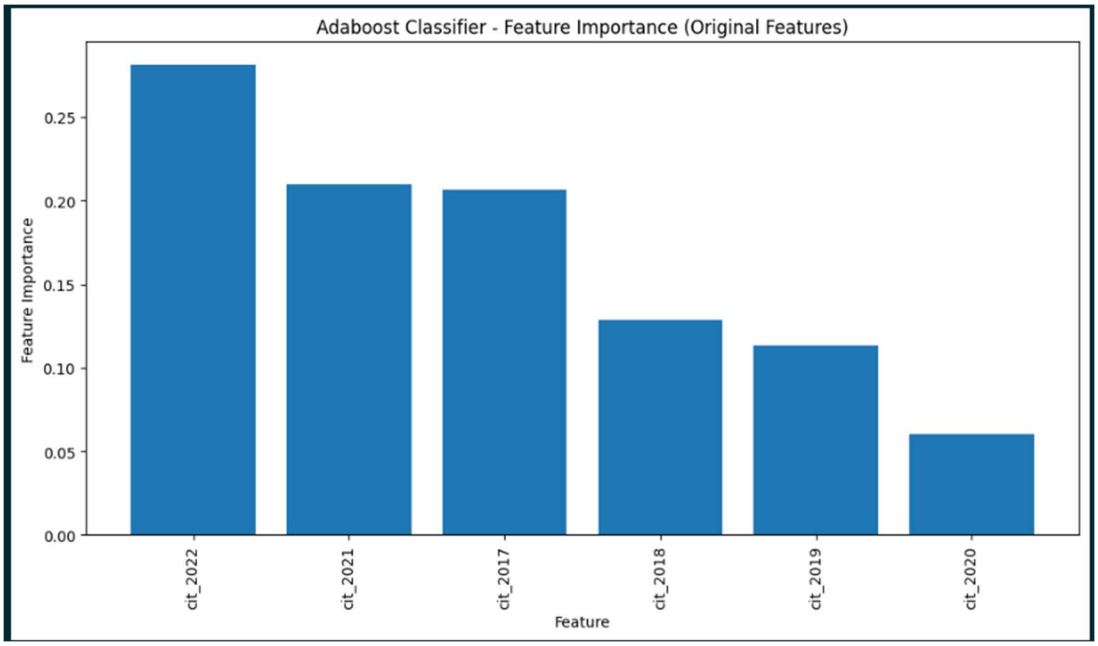
Name: Arlene Antony D'costa
Seating Pin: 79
SUID: 594303899

Display feature importance for the Adaboost model

```
feature_importances = best_clf.feature_importances_
sorted_indices = np.argsort(feature_importances)[::-1]

# Plot feature importance
plt.figure(figsize=(12, 6))
plt.bar(range(X.shape[1]), feature_importances[sorted_indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[sorted_indices], rotation=90)
plt.xlabel("Feature")
plt.ylabel("Feature Importance")
plt.title("Adaboost Classifier - Feature Importance (Original Features)")
plt.show()
```

✓ 0.2s



As observed in the above screenshot, the adaboost classifier regards the **cit_2022** as the most **important feature** as it shows a **maximum reduction in the gini index** over all the classifiers and the boosting rounds. Boosting gives higher priority to incorrectly classified samples so that those samples can be considered first in the following round and can be classified using a correct feature. Since the classification criteria for the target label was the ratio of cit_2022/cit_2021, we can observe in the above screenshot that the classifier results are directly proportional to cit_2022 and inversely proportional to cit_2021.

Hence the accuracy and f1-score obtained are the best as opposed to other models as well as easily interpretable and understandable.