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

CIS 662 HW7 Report: Random classifier

PART 1: Use the original citation for training Random Forest 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
```

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.**

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) will be labelled as "Low".

2. Medium (1.05-1.15): Values between 1.05 (inclusive) and 1.16 (exclusive) will be labelled as "Medium".

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.

We can observe from the above logic that the ratio of 2022/2021 influences the category for the citations.

Random forests build multiple decision trees by training each tree on a different subset of the data. Bootstrap sampling involves randomly selecting samples with replacement from the original dataset to create these subsets.
Samples = sqrt(features)

To determine the number of **bootstrap aggregations or samples(n_estimators**) to be provided for the random forest classifiers for efficient training, I have written a small logic

and plotted a graph using Matplotlib. For each sample, there might be features used in samples that can be repeated in multiple samples.

Plot the effect of the number of bootstrap samples on model performance

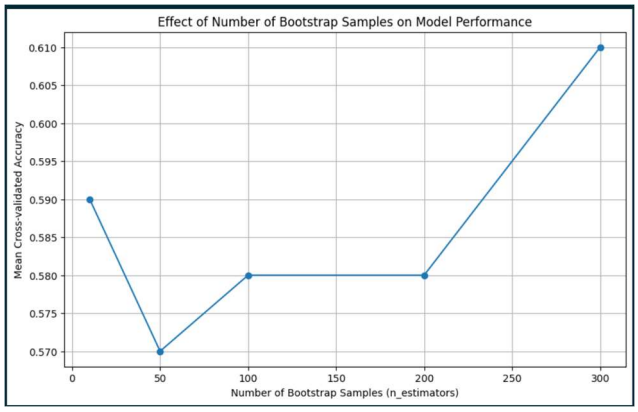
```
n_estimators_values = [10, 50, 100, 200, 300]
mean_scores = []

for n_estimators in n_estimators_values:
    rf_classifier_varying_estimators = RandomForestClassifier(n_estimators=n_estimators, random_state=20)
    scores = cross_val_score(rf_classifier_varying_estimators, X, y, cv=5, scoring='accuracy')
    mean_scores.append(np.mean(scores))

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(n_estimators_values, mean_scores, marker='o')
plt.title('Effect of Number of Bootstrap Samples on Model Performance')
plt.xlabel('Number of Bootstrap Samples (n_estimators)')
plt.ylabel('Mean Cross-validated Accuracy')
plt.grid(True)
plt.show()
```

✓ 5.3s

Here, I used 5 different sample(bagging) values and passed them as a parameter to the Random Forest Classifier.



It can be observed that Accuracy is high where bootstrap sample value is 300. Hence, I have considered the n_estimator(bagging) value as 300 for RF classifier.

OUTPUT:

RF classifier code:

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

```
n_estimators = 300

# Initialize the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=n_estimators,random_state=20)

# Train the model
rf_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_classifier.predict(X_test)
```
✓ 0.4s

Based on the above results, the Model is trained with the training dataset with sample number 300 passed as a parameter to the Random Forest classifier for training.

Output matrix:

```
Accuracy: 0.75
Classification Report:
              precision    recall  f1-score   support

        High       0.50      0.67      0.57         3
         Low       0.86      0.86      0.86        14
      Medium       0.50      0.33      0.40         3

    accuracy                           0.75        20
   macro avg       0.62      0.62      0.61        20
weighted avg       0.75      0.75      0.75        20
```

CONCLUSION:

Accuracy obtained is **75%.** Better Accuracy and F1 score can be obtained because the target variable is based on ratio of cit_2022/cit_2021, so when random classifier was trained using given original citation data, random classifier-built decision trees by trying to establish complex non-linear relationships with the target variable [ratio of 2022/2021] by calculating mean decrease in Gini index for features which can be explained better below:

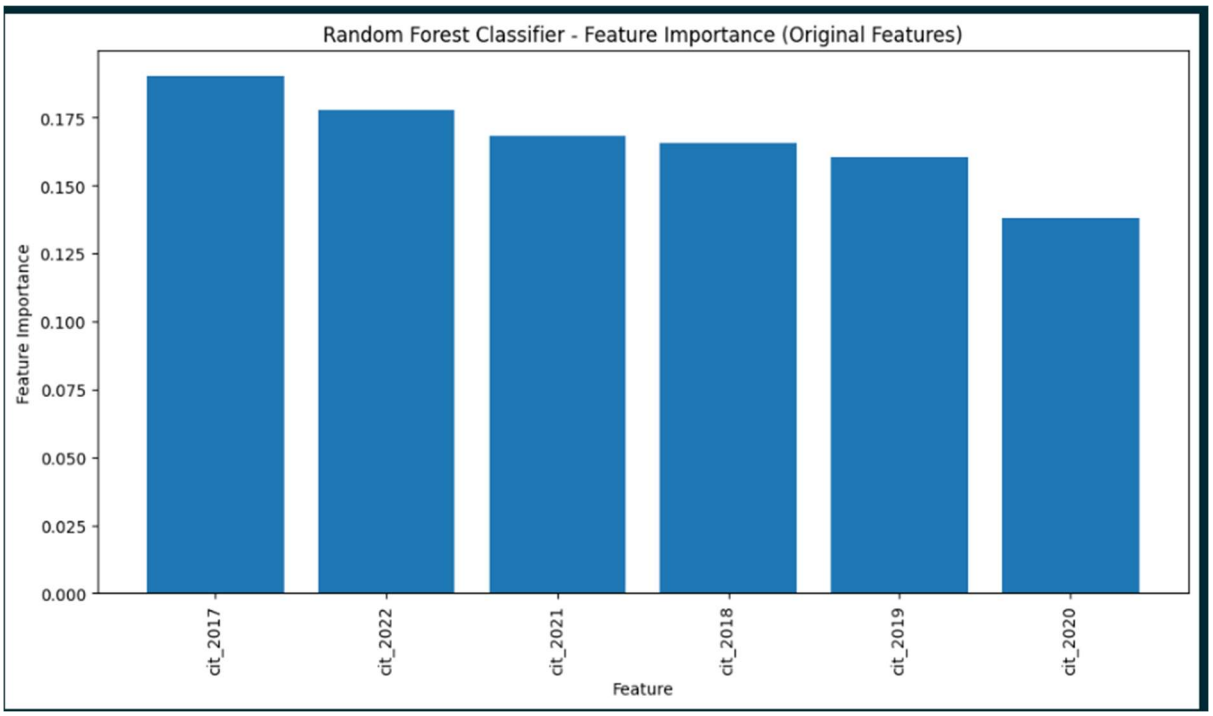
Code for identifying the most important feature selected for classification.

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

Display feature importance for the initial model

```
feature_importances = rf_classifier.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("Random Forest Classifier - Feature Importance (Original Features)")
plt.show()
```



As observed in the above screenshot, the classifier model tries to use the feature 2017 as it shows the mean decrease in Entropy value and can derive complex non-linear relationships with the target variable. Hence, it inaccurately classifies the data into 3 classes using the 2017 citation features based on spurious relationships and correlations.
As there is no direct linear/non-linear relationship of any feature variable with the target variable (ratio of cit_2022/cit_2021), the classifier randomly classifies the data as it has already established a spurious correlation with some values for feature column cit_2017.

However, the results are not bad, but they are incorrect which can be observed in the above screenshot as the target variable is the ratio between cit_2022/cit_2021 and feature cit_2017 is used majorly as the criteria for classification.

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

PART 2: Evaluating results of new features for training Random Forest Classifier

For the second part of the question, the same classifier is used for classification, but there is a slight change in the feature values.

For the new classifier instead of using cit_2017-cit_2022 as feature variables we are using the updated logic for calculating new feature variables as mentioned below:

Part 2: Random Forest Classifier with New features

Define a function to Create the new features

+ Code + Markdown

```
def calculate_new_features(df):
    for year in range(2017, 2022):
        next_year = year + 1
        feature_name = f'newfeature_{next_year}_{year}'
        data[feature_name] = (data[f'cit_{next_year}'] - data[f'cit_{year}']) / data[f'cit_{year}']
    return df
```

✓ 0.0s

```
X
```

✓ 0.1s

	newfeature_2018_2017	newfeature_2019_2018	newfeature_2020_2019	newfeature_2021_2020	newfeature_2022_2021
0	-0.048077	0.050505	-0.134615	-0.222222	-0.071429
1	0.171875	-0.160000	-0.253968	0.255319	-0.203390
2	-0.167857	-0.044349	-0.137725	0.095486	-0.028526
3	0.325581	0.403509	0.150000	-0.119565	0.000000
4	6.500000	-0.483333	2.290323	0.313725	0.507463
...	...	...	...	...	...
95	0.002262	-0.088036	-0.121287	0.084507	-0.135065
96	0.191278	0.165061	0.079383	-0.016343	-0.051402
97	0.054767	-0.134615	0.053333	-0.130802	-0.179612
98	-0.189189	0.700000	0.039216	0.264151	-0.208955
99	8.500000	4.000000	2.400000	2.029412	0.514563

100 rows × 5 columns

As shown in the above screenshot the formula for calculating the new feature:

New_feature = next_year – curr_year / curr_year

In the below screenshot, for the values where citations are 0 the ratio turns out to be **NaN** and **inf**.

```
datanew.loc[[91], :]
```

✓ 0.0s Python

	cit_2021	cit_2022	ratio_21_22	category	newfeature_2018_2017	newfeature_2019_2018	newfeature_2020_2019	newfeature_2021_2020	newfeature_2022_2021
2	63	121	1.92	High	NaN	inf	3.4	1.863636	0.920635

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

I have handled such data by replacing the **-inf** value with a value lower than **the min** value and the **+inf** value with a value higher than **the max** value.

Handling Missing and Nan values(Manual // Can be done via automation)

+ Code + Markdown

```
#X.loc[:, X.isin([np.inf, -np.inf])] = X.replace([np.inf, -np.inf], [8.5, -66])
# Assuming X is a Pandas DataFrame
columns_with_inf = X.columns[X.isin([np.inf, -np.inf]).any()]

# Replace infinite values by creating a copy
X.loc[:, columns_with_inf] = X[columns_with_inf].copy().replace([np.inf, -np.inf], [8.5, -66])
X.fillna(0, inplace=True)
```

✓ 0.3s

```
nan_rows = X[X.isna().any(axis=1)]
```

✓ 0.0s

+ Code + Markdown

```
nan_rows
```

✓ 0.0s

newfeature_2018_2017	newfeature_2019_2018	newfeature_2020_2019	newfeature_2021_2020	newfeature_2022_2021

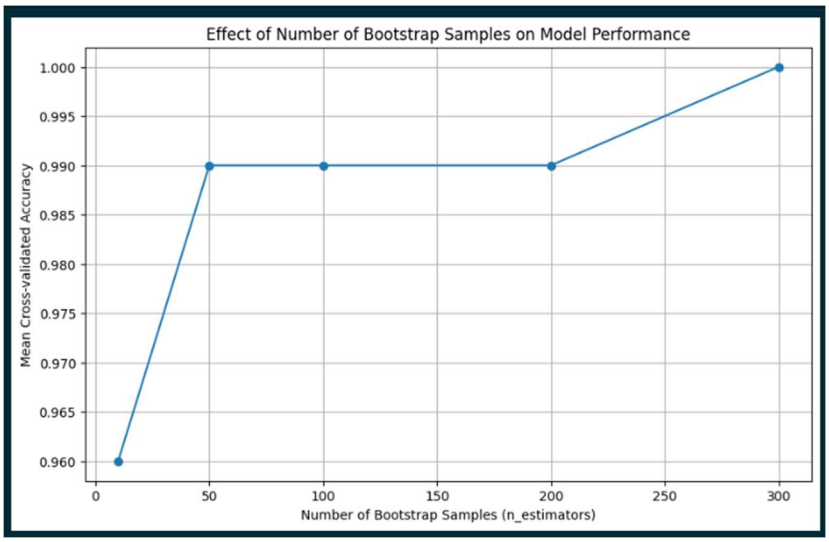
I have used the same logic as mentioned in part 1 for estimating the number of bootstrap samples to be passed as a parameter to Random Forest classifier.

```
n_estimators_values = [10, 50, 100, 200, 300]
mean_scores = []

for n_estimators in n_estimators_values:
    rf_classifier_varying_estimators = RandomForestClassifier(n_estimators=n_estimators, random_state=20)
    scores = cross_val_score(rf_classifier_varying_estimators, X, y, cv=5, scoring='accuracy')
    mean_scores.append(np.mean(scores))

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(n_estimators_values, mean_scores, marker='o')
plt.title('Effect of Number of Bootstrap Samples on Model Performance')
plt.xlabel('Number of Bootstrap Samples (n_estimators)')
plt.ylabel('Mean Cross-validated Accuracy')
plt.grid(True)
plt.show()
```

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



Even for the new dataset with the new features, Accuracy score is higher with bootstrap sample size of 300.

Dataset is split into training and testing set in the ratio 80:20.

OUTPUT:

Train the Random Forest Classifer model

```
n_estimators = 300

# Initialize the Random Forest classifier
rf_classifier_new_features = RandomForestClassifier(n_estimators=n_estimators,random_state=20)

# Train the model
rf_classifier_new_features.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_classifier.predict(X_test)
```

✓ 2.9s

Output matrix:

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

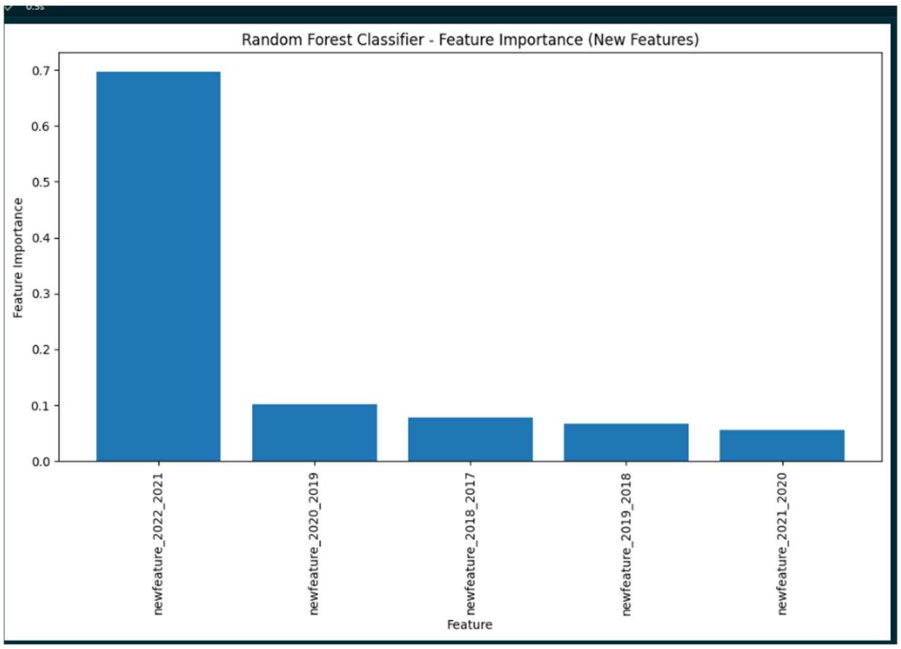
```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

        High       1.00      1.00      1.00         3
         Low       1.00      1.00      1.00        14
      Medium       1.00      1.00      1.00         3

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

CONCLUSION:

Accuracy obtained is 100% for the testing set of the new dataset.



Since the target variable is obtained on the ratio of cit_2022/cit_2021 and the new dataset featured column containing the ratio **2022-2021/2021** is closer to the target variable as it shows **mean decrease in entropy/Gini index value** and since Random Forest classifier works by detecting non-linear patterns and correlations between the feature columns and the target variable, the model was able to accurately classify the test sample into the appropriate classes giving the accuracy and F1 score of 100%.