LAB 2: DECISION TREE (NURSERY DATASET)

CS420 – Artificial Intelligence

Student: Phan Thanh An

Student's ID: 21127003

Class: 21CLC07 – Term III/2023

AUGUST 24, 2023
HCMUS
Ho Chi Minh City

Contents

I. General 3

II. Preparing dataset 4

III. Building the decision tree classifiers 6

1. Standardizing classnames and features name 6
2. Build decision tree using DecisionTreeClassifier 6
3. Export PDF using graphviz 7

IV. Evaluating the decision tree classifiers 9

1. Classification report and confusion matrix 9
2. Comments on each decision tree 9

a. Decision tree classifer 40/60 10

b. Decision tree classifer 60/40 11

c. Decision tree classifer 80/20 12

d. Decision tree classifer 90/10 13

e. Overall 14

V. The depth and accuracy of a decision tree 15

1. Accuracy Score and Max Depth 15
2. Comment on the statistic 15

VI. References 17

IMAGE CONTENT

Image 1. Decision tree example....3
Image 2. Visulization of (40/60) subset....5
Image 3. Classification criteria parameter....7
Image 4. Classification report of (40/60) subset....10
Image 5. Classification report of (60/40) subset....11
Image 6. Classification report of (80/20) subset....12
Image 7. Classification report of (90/10) subset....13
Image 8. Accuracy of different depth....15

I. General

A decision tree is a visual representation and a simple, yet powerful machine learning algorithm used for making decisions or predictions. It mimics the way humans make decisions by breaking down a complex problem into a series of simpler, binary choices.

The decision tree is like a flowchart for decision-making, where you start with a question or feature, make binary choices at each node, and eventually reach a final decision or prediction at the leaves. It's a valuable tool in machine learning and data analysis for tasks ranging from customer churn prediction to medical diagnosis.

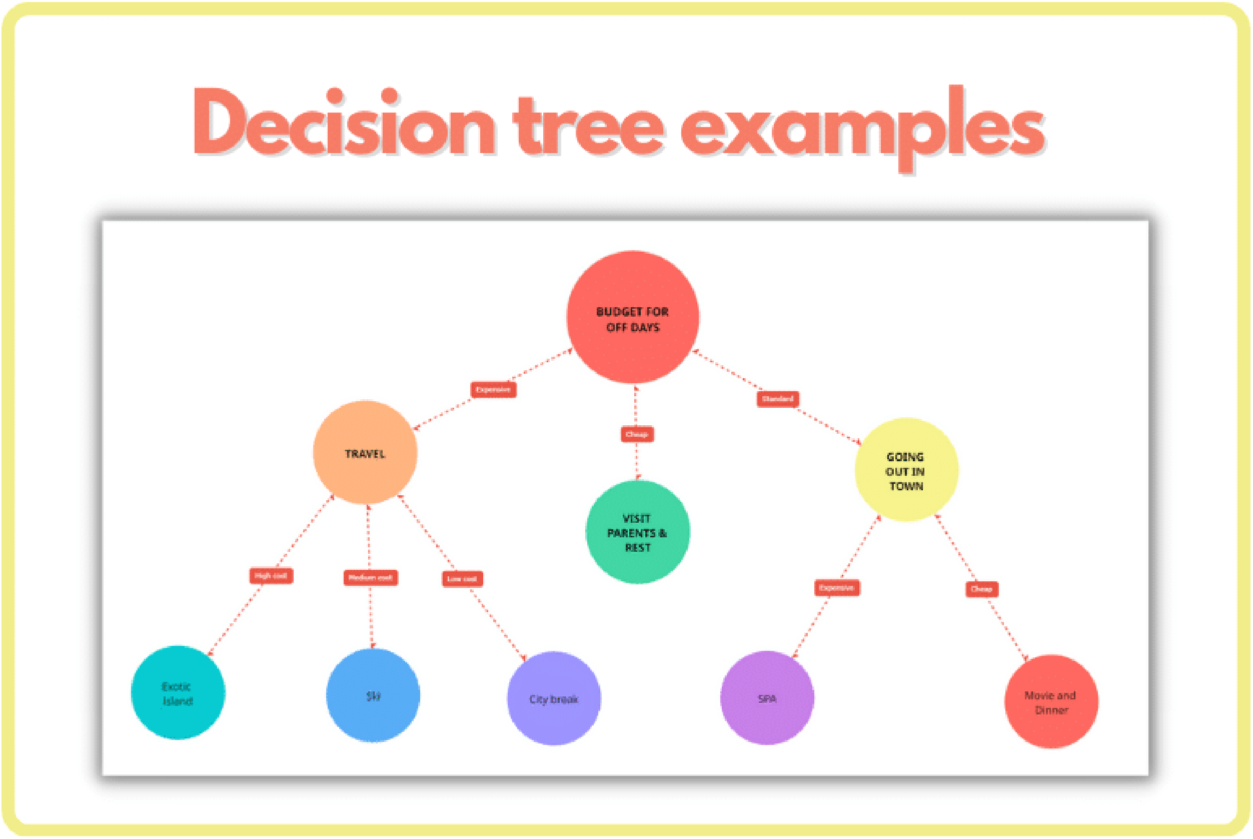


Image 1. Decision tree example

II. Preparing dataset

Reads a CSV file containing data, applies label encoding to convert categorical data into numerical format, and then separates the features and labels for further analysis or machine learning tasks.

```python
datasets = pd.read_csv(f'data/nursery.csv', header=None)
datasets = datasets.apply(LabelEncoder().fit_transform)

original_features = datasets.iloc[:, :-1]
original_labels = datasets[datasets.columns[-1]]
```

- **features**: This argument represents the input features of dataset. These could be various attributes or measurements that you use as input for your machine learning model.

- **labels**: This argument represents the corresponding labels or targets associated with each set of features. In a supervised machine learning context, these labels represent the expected outputs that the model should learn to predict.

- **ratios**: This argument is a list containing two values. The first value ratios[0] represents the ratio of the dataset that should be used for training, and the second value ratios[1] represents the ratio of the dataset that should be used for testing.

```python
def make_subset(features, labels, ratios):
    # (train_features, test_features, train_labels, test_labels)
    subset = train_test_split(
        features,
        labels,
        test_size=ratios[1],
        train_size=ratios[0],
        random_state=42,
        shuffle=True
    )

    return subset
```

With each subset we use ***make_subset*** method to create a subset with ***ratios=(x, y)*** while **x** is percentage for training set and **y** is percentage for testing set. For example, here is an implementation for making **(40/60)** subset and push it to subsets list.

```
    ratios = [0.4, 0.6]
    subset = make_subset(original_features, original_labels, ratios)
    subsets.append(subset)
```

After a subset was made, we can visualize it with comparision for original, training and testing datasets.

```
    visualize_subset(subset, title="(40/60) subset")
```

*(*View all other visualization in jupyter notebook)*

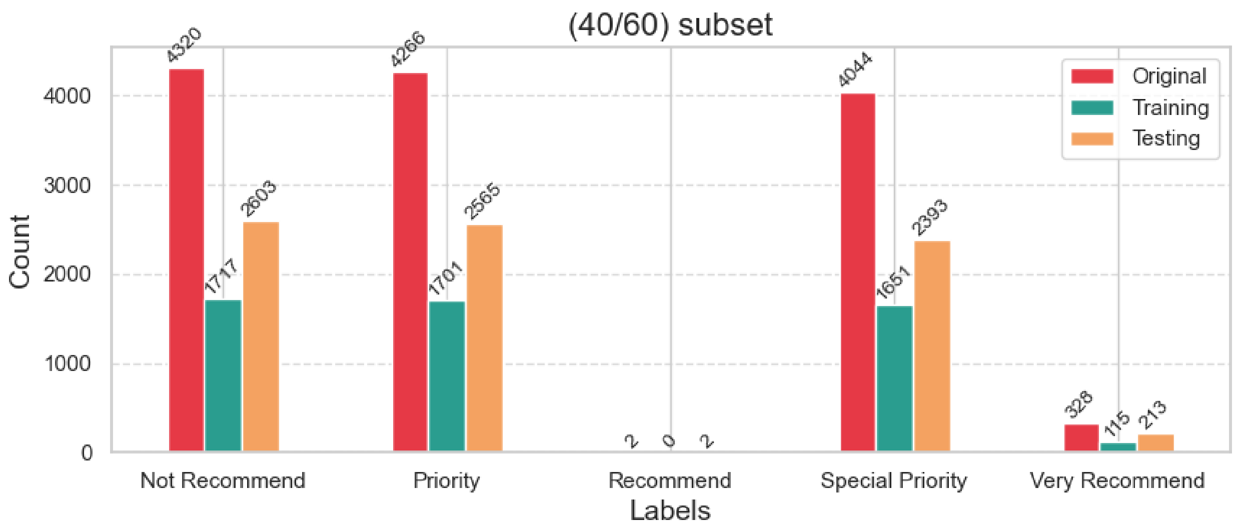


Image 2. Visulization of (40/60) subset

III. Building the decision tree classifiers

1. Standardizing classnames and features name

Based on the attributes value was described in nursery.data we redefined features_names and class_names for building decision tree.

Attribute Values:

• parents	usual, pretentious, great_pret
• has_nurs	proper, less_proper, improper, critical, very_crit
• form	complete, completed, incomplete, foster
• children	1, 2, 3, more
• housing	convenient, less_conv, critical
• finance	convenient, inconv
• social	non-prob, slightly_prob, problematic
• health	recommended, priority, not_recom

```
feature_names=["parent", "has_nurs", "form", "children", "housing", "finance",
"social", "health"]
class_names = ["Not Recommend", "Priority", "Recommend", "Special Priority", "Very
Recommend"]
```

2. Build decision tree using DecisionTreeClassifier

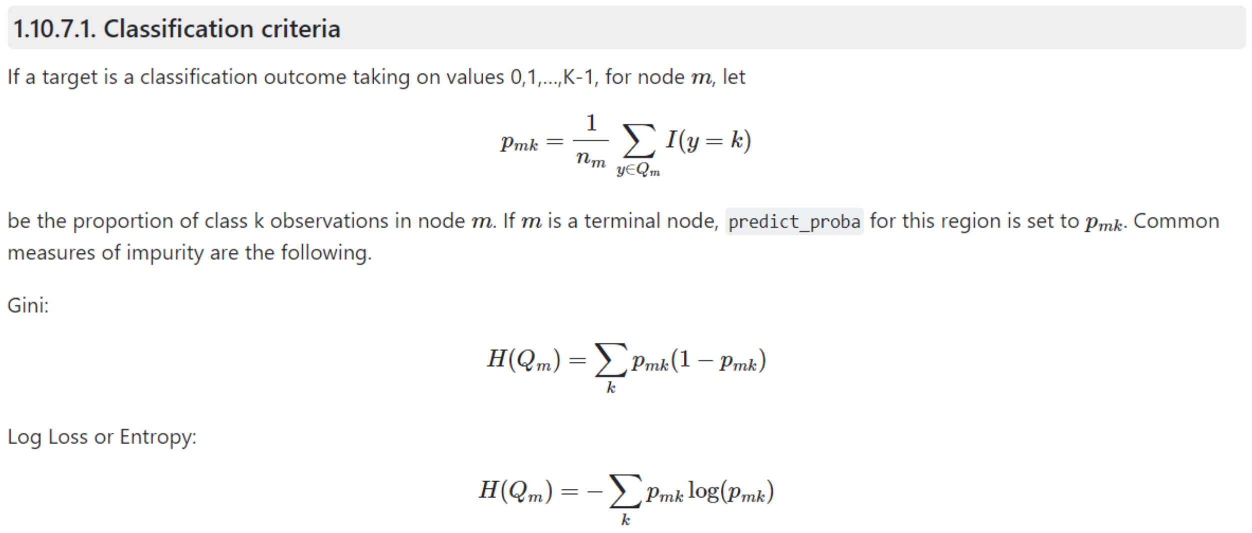
DecisionTreeClassifier is the class from scikit-learn used for building decision tree models.

```
feature_train, feature_test, label_train, label_test = subsets[0]
trees[0] = DecisionTreeClassifier(
    criterion = 'entropy',
    random_state = 0,
    max_depth = 7
)
trees[0].fit(feature_train, label_train.ravel())
```

The criterion parameter specifies the function used to measure the quality of splits in the decision tree. In this case, "entropy" is used as the criterion. Entropy is a measure of impurity in a set of labels. The decision tree algorithm will try to minimize entropy at each node during the tree-building process.

Here is a description of scikit-learn about criterion parameters.

1.10. Decision Trees — scikit-learn 1.3.0 documentation (click here)



1.10.7.1. Classification criteria

If a target is a classification outcome taking on values 0,1,...,K-1, for node m, let

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$$

be the proportion of class k observations in node m. If m is a terminal node, `predict_proba` for this region is set to p_{mk}. Common measures of impurity are the following.

Gini:

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$$

Log Loss or Entropy:

$$H(Q_m) = - \sum_k p_{mk} \log(p_{mk})$$

Image 3. Classification criteria parameter

random_state=0 sets a seed for the random number generator used internally by the decision tree algorithm. Setting this parameter ensures that the randomness in the algorithm's operations is reproducible. If you use the same random_state value in the future, you will get the same results, making your experiments more consistent.

.fit(feature_train, label_train.ravel()) calls the .fit() method on the decision tree classifier. This method trains the model using the provided training data.

3. Export PDF using graphviz

```
data = tree.export_graphviz(
    trees[1],
    out_file = None,
    feature_names = feature_names,
    class_names = class_names,
    filled = True,
    max_depth = 7,
    special_characters = True
)
graph = graphviz.Source(data)
```

```
graph.render("file_name", format = "pdf", cleanup=True)
```

tree.export_graphviz is part of the scikit-learn library (sklearn.tree) and is used to export a decision tree in DOT format. The DOT format is a plain text graph description language that represents the decision tree's structure.

graphviz.Source function is used to create a graph from the DOT data. This function is part of the Graphviz library, which is used for visualizing graph data structures.

graph.render is called to render and save the graph as a PDF file. The first argument specifies the file path and name where the PDF will be saved.

*(*You can see all decision tree which was exported in **decision_tree_exported** folder)*

IV. Evaluating the decision tree classifiers

1. Classification report and confusion matrix

```
class_names = [
    "Not Recommend",
    "Priority",
    "Recommend",
    "Special Priority",
    "Very Recommend"
]
_, feature_test, _, label_test = subsets[0]
label_pred = trees[0].predict(feature_test)
print(classification_report(
    label_test,
    label_pred,
    target_names = class_names,
    zero_division = 1)
)
```

zero_division=1 handles the scenario where there might be a division by zero when calculating metrics. If a class has no true positives, true negatives, or false positives, it can result in division by zero when calculating precision, recall, or F1-score. By setting zero_division to 1, it ensures that no division by zero errors occurs and sets the relevant metric (precision, recall, or F1-score) to 1 for such cases.

The **classification_report** function from scikit-learn is commonly used to generate a comprehensive report of the classification performance of a machine learning model. It provides various metrics such as **precision**, **recall**, **F1-score**, and **support** for each class. In this case, the classification_report is being used to report on the performance of a model's predictions made with label_test and label_pred.

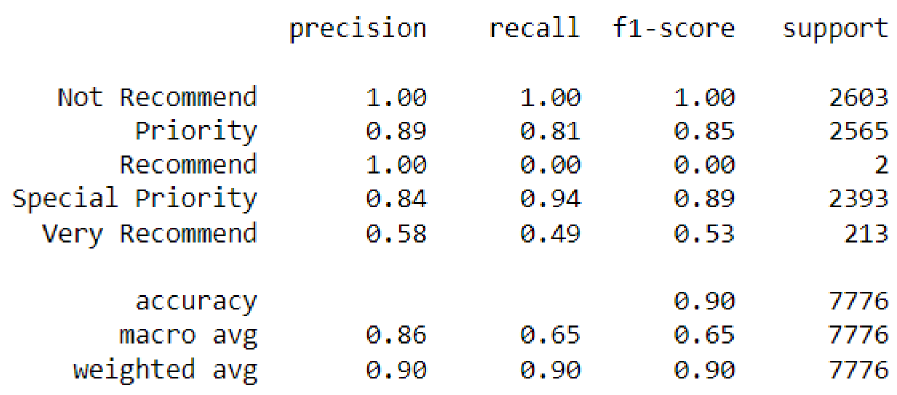
2. Comments on each decision tree

Firstly, we must know metrics such as **precision**, **recall**, **F1-score, accuracy**, and **support,** what does it mean?

- **Precision** measures the accuracy of the positive predictions made by the model for each class. It answers the question, "Of all the instances that the model predicted as class X, how many were correctly classified as class X?"

- **Recall** measures the ability of the model to identify all the relevant instances of each class. It answers the question, "Of all the instances that are actually class X, how many did the model correctly identify as class X?"
- **F1-Score** is the harmonic mean of precision and recall. It provides a balance between precision and recall. It's a useful metric when you want to consider both false positives and false negatives.
- **Support** indicates the number of instances for each class in the test set
- **Accuracy** considers the correct predictions across all classes in the test dataset. Support values for individual classes do not directly impact overall accuracy, but they provide context for the accuracy score.

a. Decision tree classifer 40/60



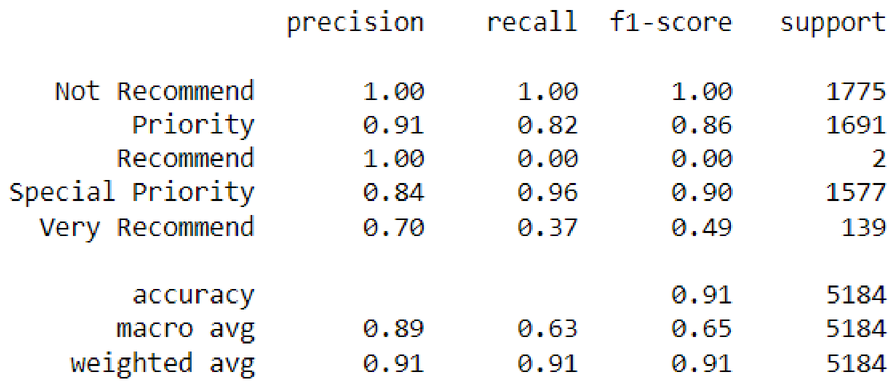
```
                  precision    recall  f1-score   support

   Not Recommend       1.00      1.00      1.00      2603
        Priority       0.89      0.81      0.85      2565
       Recommend       1.00      0.00      0.00         2
Special Priority       0.84      0.94      0.89      2393
  Very Recommend       0.58      0.49      0.53       213

        accuracy                           0.90      7776
       macro avg       0.86      0.65      0.65      7776
    weighted avg       0.90      0.90      0.90      7776
```

Image 4. Classification report of (40/60) subset

- **High Accuracy**: The overall accuracy of 90% suggests that the model performs reasonably well on the dataset. It correctly classifies a large portion of instances.
- **Not Recommend**: The precision, recall, and F1-score for the "Not Recommend" class are all perfect (1.00), indicating that the model excels in correctly identifying instances of this class. The support value of 2603 shows that there is a substantial number of instances in this class.
- **Special Priority**: The model also performs well for the "Special Priority" class with a precision of 0.84 and a high recall of 0.94. The F1-score of 0.89 indicates a good balance between precision and recall for this class.

- **Priority**: The model exhibits decent precision (0.89) for the "Priority" class but struggles with recall (0.81), meaning it misses some instances of this class. This is evident in the F1-score of 0.85, which is lower compared to "Not Recommend" and "Special Priority." Further investigation might be needed to improve the performance of this class.
- **Very Recommend**: The model performs less well on the "Very Recommend" class with a precision of 0.58 and recall of 0.49. The F1-score of 0.53 indicates room for improvement in correctly classifying instances of this class. The support value of 213 suggests that this class is underrepresented, which might contribute to the lower performance.
- **Recommend**: The "Recommend" class has a support of only 2 instances, making it challenging to assess the model's performance accurately. Precision and F1-score are both low (0.00), indicating that the model struggles to correctly identify this class. However, this class might require further investigation due to its extremely low representation.

b. Decision tree classifer 60/40



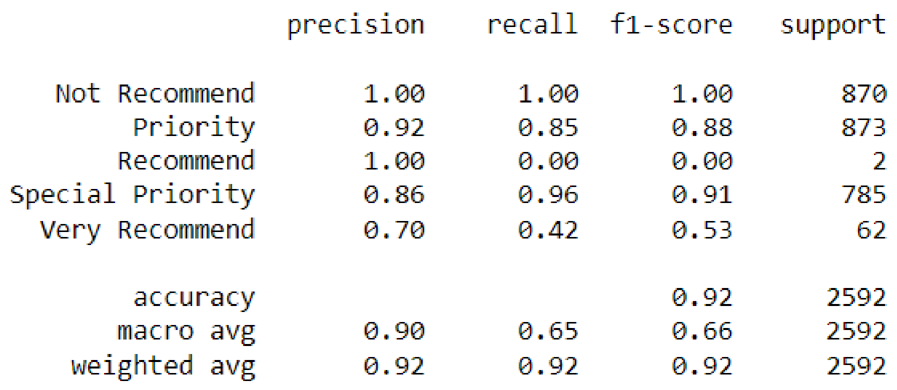
```
                  precision    recall  f1-score   support

   Not Recommend       1.00      1.00      1.00      1775
        Priority       0.91      0.82      0.86      1691
       Recommend       1.00      0.00      0.00         2
Special Priority       0.84      0.96      0.90      1577
  Very Recommend       0.70      0.37      0.49       139

        accuracy                           0.91      5184
       macro avg       0.89      0.63      0.65      5184
    weighted avg       0.91      0.91      0.91      5184
```

Image 5. Classification report of (60/40) subset

- **High Accuracy**: The overall accuracy of 91% indicates that the model performs well on the dataset, correctly classifying a large portion of instances.
- **Not Recommend**: The model demonstrates perfect precision, recall, and F1-score (all 1.00) for the "Not Recommend" class. This suggests that the model excels in correctly identifying instances of this class. The support value of 1775 indicates a substantial number of instances in this class.

- **Special Priority**: The model performs well for the "Special Priority" class with a precision of 0.84 and a high recall of 0.96. The F1-score of 0.90 indicates a good balance between precision and recall for this class.
- **Priority**: The model demonstrates solid precision (0.91) and a respectable recall (0.82), indicating that it correctly identifies most instances of this class. The F1-score of 0.86 suggests a good overall balance between precision and recall.
- **Very Recommend**: The model faces difficulties in correctly classifying the "Very Recommend" class, with a precision of 0.70 and a recall of 0.37. The F1-score of 0.49 indicates room for improvement in correctly classifying instances of this class. The support value of 139 suggests that this class is underrepresented, which may contribute to the lower performance.
- **Recommend**: Like the (40/60 decision tree, the "Recommend" class has a support of only 2 instances, making it challenging to assess the model's performance accurately. Precision and F1-score are both low (0.00), indicating that the model struggles to correctly identify this class. As previously mentioned, this class might require further investigation and potentially more data for reliable classification.

c. **Decision tree classifer 80/20**



```
                 precision    recall  f1-score   support

   Not Recommend       1.00      1.00      1.00       870
        Priority       0.92      0.85      0.88       873
       Recommend       1.00      0.00      0.00         2
Special Priority       0.86      0.96      0.91       785
  Very Recommend       0.70      0.42      0.53        62

        accuracy                           0.92      2592
       macro avg       0.90      0.65      0.66      2592
    weighted avg       0.92      0.92      0.92      2592
```

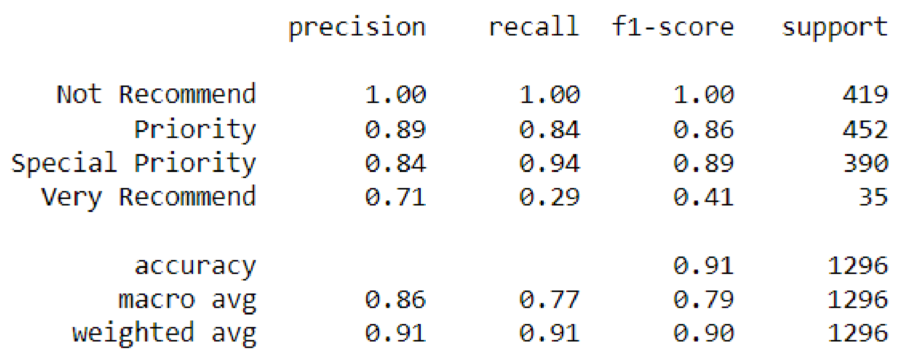
Image 6. Classification report of (80/20) subset

- High Accuracy: The overall accuracy of 92% indicates that the model performs very well on the dataset, correctly classifying a significant portion of instances.
- Not Recommend: The model demonstrates perfect precision, recall, and F1-score (all 1.00) for the "Not Recommend" class. This suggests that the model excels in correctly

identifying instances of this class. The support value of 870 indicates a substantial number of instances in this class.

- Special Priority: The model performs well for the "Special Priority" class with a precision of 0.86 and a high recall of 0.96. The F1-score of 0.91 indicates a good balance between precision and recall for this class.
- Priority: The model demonstrates solid precision (0.92) for the "Priority" class and a respectable recall (0.85), indicating that it correctly identifies most instances of this class. The F1-score of 0.88 suggests a good overall balance between precision and recall.
- Very Recommend: The model faces difficulties in correctly classifying the "Very Recommend" class, with a precision of 0.70 and a recall of 0.42. The F1-score of 0.53 indicates room for improvement in correctly classifying instances of this class. The support value of 62 suggests that this class is underrepresented, which may contribute to the lower performance.
- Recommend: Similar to the previous reports, the "Recommend" class has support of only 2 instances, making it challenging to assess the model's performance accurately. Precision and F1-score are both low (0.00), indicating that the model struggles to correctly identify this class. As previously mentioned, this class might require further investigation and potentially more data for reliable classification.

d. Decision tree classifer 90/10



```
                  precision    recall  f1-score   support

   Not Recommend       1.00      1.00      1.00       419
        Priority       0.89      0.84      0.86       452
Special Priority       0.84      0.94      0.89       390
  Very Recommend       0.71      0.29      0.41        35

        accuracy                           0.91      1296
       macro avg       0.86      0.77      0.79      1296
    weighted avg       0.91      0.91      0.90      1296
```

Image 7. Classification report of (90/10) subset

- **High Accuracy**: The overall accuracy of 91% indicates that the model performs well on the dataset, correctly classifying a significant portion of instances.

- **Not Recommend:** The model demonstrates perfect precision, recall, and F1-score (all 1.00) for the "Not Recommend" class. This suggests that the model excels in correctly identifying instances of this class. The support value of 419 indicates a substantial number of instances in this class.
- **Special Priority:** The model performs well for the "Special Priority" class with a precision of 0.84 and a high recall of 0.94. The F1-score of 0.89 indicates a good balance between precision and recall for this class.
- **Priority**: The model demonstrates solid precision (0.89) for the "Priority" class and a respectable recall (0.84), indicating that it correctly identifies most instances of this class. The F1-score of 0.86 suggests a good overall balance between precision and recall.
- **Very Recommend**: The model faces difficulties in correctly classifying the "Very Recommend" class, with a precision of 0.71 and a recall of 0.29. The F1-score of 0.41 indicates room for improvement in correctly classifying instances of this class. The support value of 35 suggests that this class is underrepresented, which may contribute to lower performance.

e. Overall

In summary, the model demonstrates high accuracy and excels in some classes, particularly "Not Recommend," "Priority," and "Special Priority." However, it faces challenges with the "Very Recommend" class and continues to struggle with the "Recommend" class due to extremely low support. Further tuning and consideration of class imbalances may lead to improvements in overall performance.

V. The depth and accuracy of a decision tree

*(*You can see decision tree with different depth in* ***decision_tree_with_depth*** *folder)*

1. Accuracy Score and Max Depth

max_depth	**None**	**2**	**3**	**4**	**5**	**6**	**7**
Accuracy	0.995	0.77	0.815	0.86	0.876	0.891	0.922

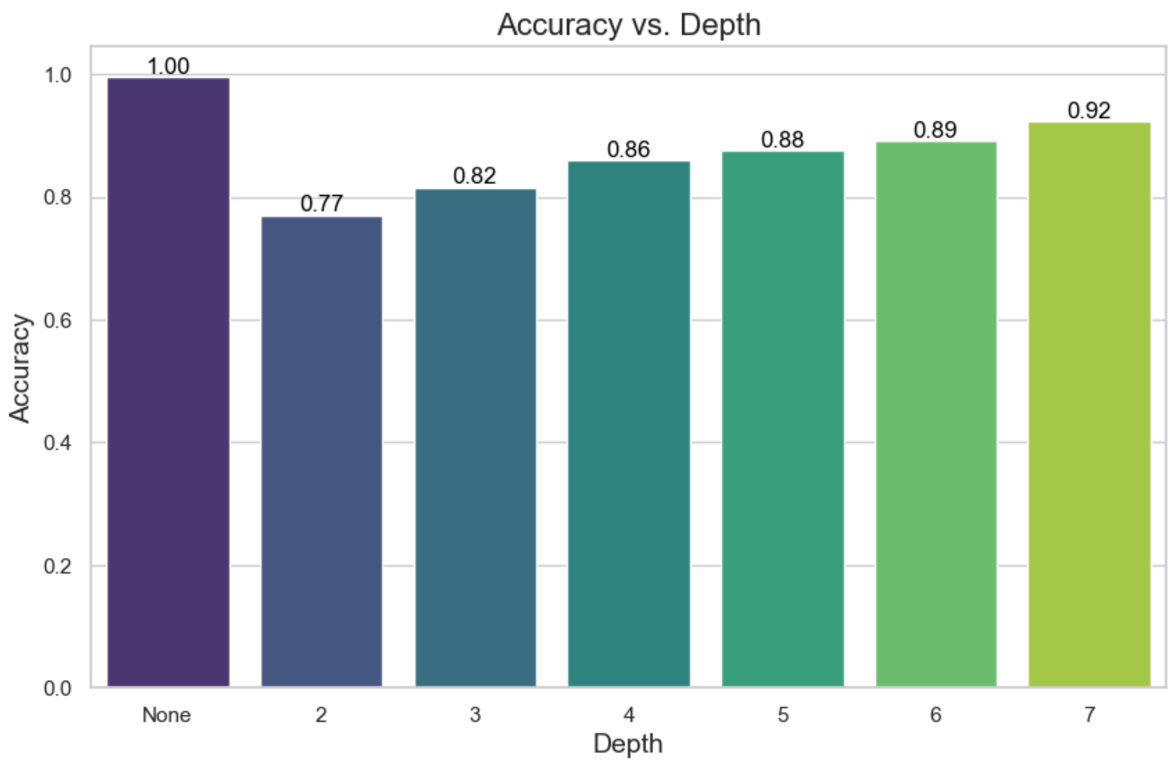


Image 8. Accuracy of different depth

2. Comment on the statistic

Max Depth = none with accuracy = 95%: When the maximum depth is not limited, the decision tree can grow until it perfectly fits the training data. This extreme accuracy suggests potential overfitting, where the model captures noise in the training data, making it less generalizable for new, unseen data. While the training data is perfectly learned, the model may not perform well on new data.

Max Depth = 2 with accuracy = 77%: With a maximum depth of 2, the decision tree is quite shallow. It struggles to capture complex relationships in the data, resulting in lower accuracy.

This suggests underfitting, where the model is too simple to represent the underlying patterns in the data.

Max Depth = 3 wih accuracy = 81.5%: increasing the maximum depth to 3 improves accuracy slightly. The model becomes slightly more capable of capturing patterns in the data, but it may still struggle with more intricate relationships.

Max Depth = 4 with accuracy = 86%: At a maximum depth of 4, the model's accuracy continues to improve. It can capture more complex patterns in the data, and the performance is getting closer to a good balance between underfitting and overfitting.

Max Depth = 5 and 6 with 87.6% and 89.1%: As the maximum depth increases to 5 and 6, the model becomes more accurate, indicating that it can handle even more complex relationships in the data. However, there might be a point of diminishing returns in terms of complexity and accuracy.

Max Depth = 7 with accuracy: 92.2%: The highest accuracy is achieved at a maximum depth of 7. At this depth, the model is likely providing a good balance between capturing complex patterns in the data and avoiding overfitting. This depth seems to generalize well to the test data.

→ *These results emphasize the importance of hyperparameter tuning, especially adjusting the maximum depth, to optimize the decision tree's performance on a specific dataset. Grid search, cross-validation, or other techniques can help identify the best depth for a given problem.*

VI. References

[1] Graphviz

[2] 1.10. Decision Trees — scikit-learn 1.3.0 documentation

[3] Classification And Regression Trees for Machine Learning

[4] Nursery_Classification: INSE6180 Data Mining Project (github.com)

[5] How to Calculate Precision, Recall, F1, and More for Deep Learning Models

[6] Tree Based Algorithms | Implementation In Python & R (analyticsvidhya.com)