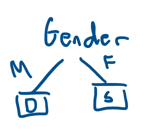
Assignment 1.4

Thursday, February 1, 2024 6:35 PM

4. I a)



Using code in q_4.py, 81 out of 314 women did not survive and 109 out of 577 men surivered

$\therefore$ impurity $= 81/(314+577) + 109/(314+577)$

$= 21.324\%$ → percentage of not correct classification

b) There are multiple ways to split by a numerical value (i.e, domain of $x \in [0,100]$, can choose any value between 0 and 100 to split), impurity becomes an important factor when selecting the threshold to split a numerical variable (want to minimize impurity with chosen threshold)

c) Code in q_4.py

Splitting on age ≤25: 255 youngs died, 197 old survived

impurity: $\frac{255+197}{314+577} = 0.4736 = 47.36\%$

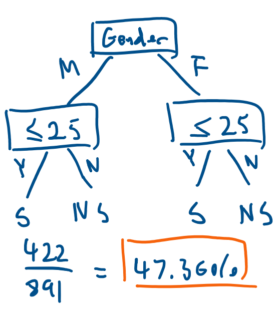
Splitting on age ≥65: 539 young died, 1 old survived

impurity: $\frac{539+1}{314+577} = 0.606060 = 60.6\%$

Since splitting age at ≤25 leads to less impurity, it results in a better decision tree

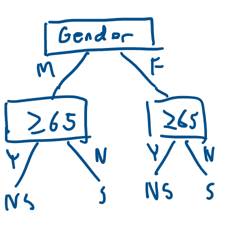
d) Split: Gender → ≤25

Code in q_4.py



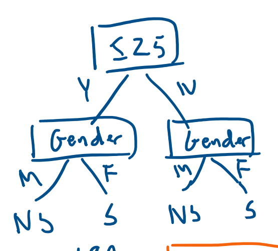
impurities: $\frac{422}{891} = 47.36\%$

Split: Gender → ≥65



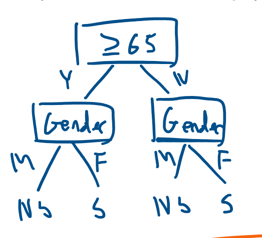
impurities: $\frac{540}{891} = 60.6\%$

Split: ≤25 → Gender



impurities: $\frac{190}{891} = 21.324\%$

Split: ≥65 → Gender



impurities: $\frac{190}{891} = 21.324\%$

The impurities are the same as the impurities calculated in parts a) and c). This is because the decision of whether someone survives is solely based on the second feature. Thus, it does not matter what the split is with the first feature, the impurities of the model is soldy dependent on the split of the second feature.

e) code in q_4.py

i) To get the Gini index, I first split up the data by the values of a feature. Then I calculated the probabilities of survive and not survive for that value. The Gini index is then calculated for each value, and final weighted Gini index is found by:

$$\text{Weighted Gini} = \sum \frac{\#\text{ of passenger with specific feature}}{\text{total }\#\text{ of passenger}} \times \text{Gini index of specific feature}$$

(ie: Male/Femal : age: 21, 24, etc)

ie : $\frac{\#\text{ of male}}{\#\text{ of passenger}} \times Gini_{male} + \frac{\#\text{ of Female}}{\#\text{ of passenger}} \times Gini_{female}$

$Gini_{male} = 1 - (P(\text{male survive})^2 + P(\text{male not survive})^2)$

Result: Weighted Gini of Gender: 0.333

Weighted Gini of Age: 0.417

Since Gini index of Gender is lower, the split is better and more pure

ii) The Shannon Entropy was found in the same way as the Gini index, except with a different equation to measure impurities:

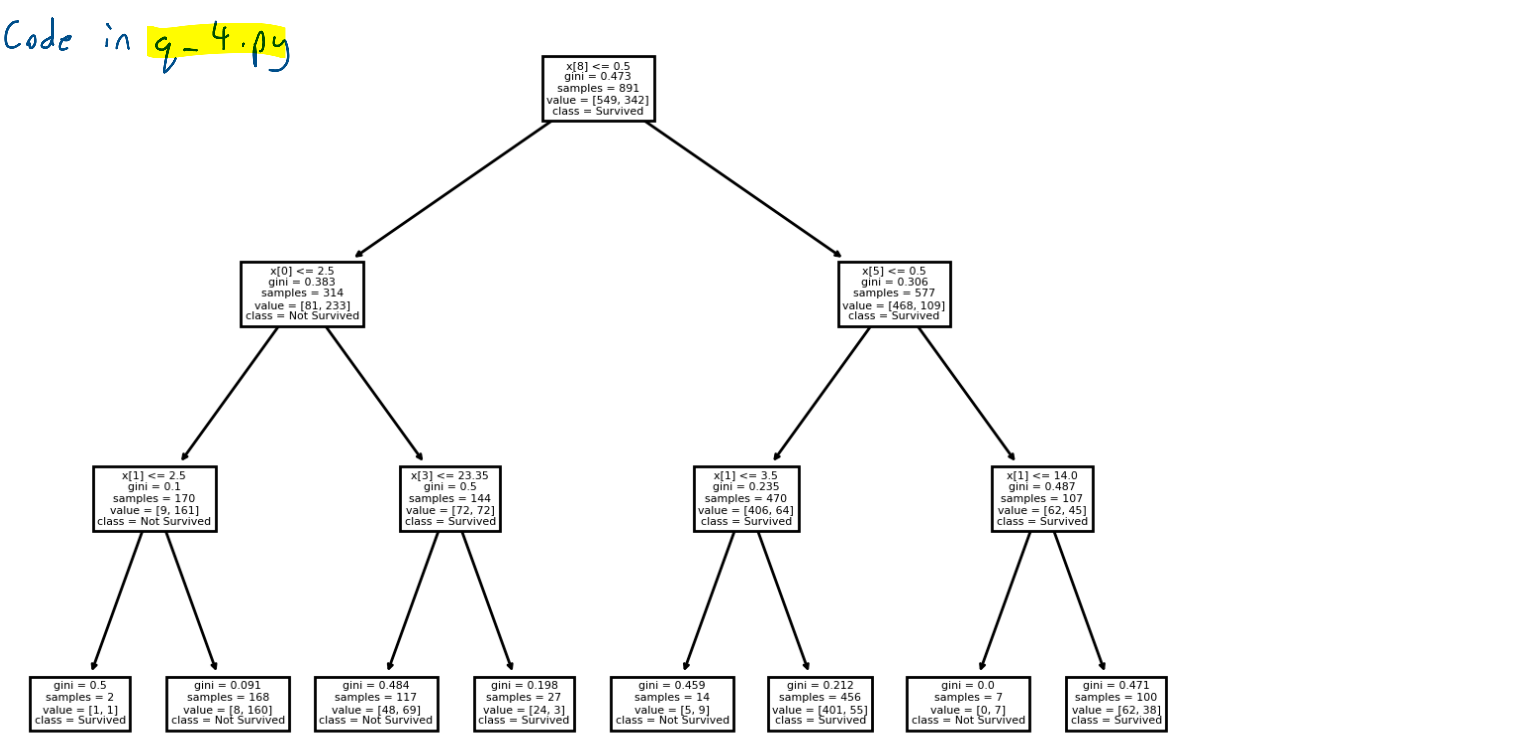
$$H(x) = -\sum_{i=1}^{n} P_i \log_2 P_i$$

Results: Weighted Shannon Entropy of Gender: 0.7405

Weighted Shannon Entropy of Age: 0.8639

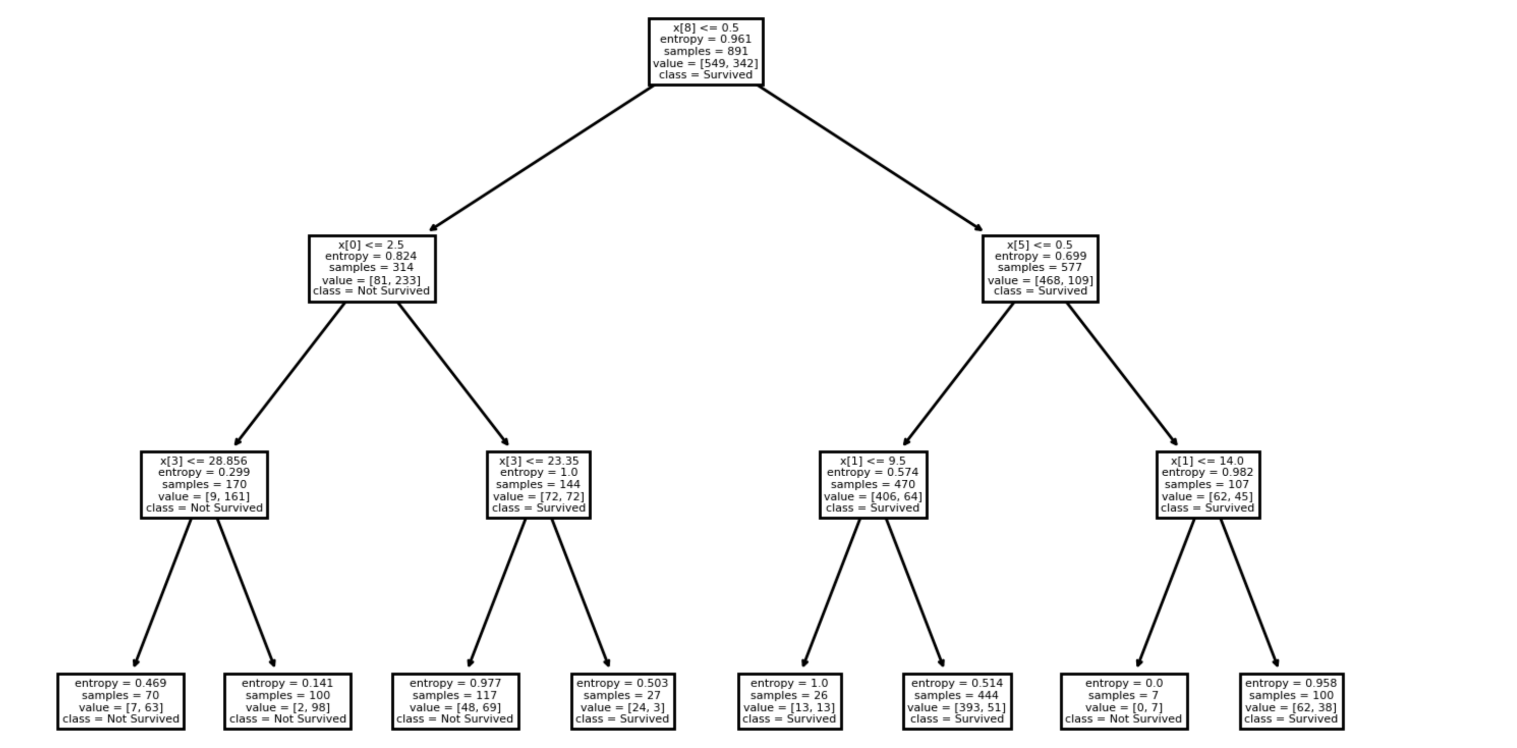
Shannon Entropy shows that Gender feature is more pure and will get a better split, which matches with the results of the Gini Index

II a) Code in q_4.py



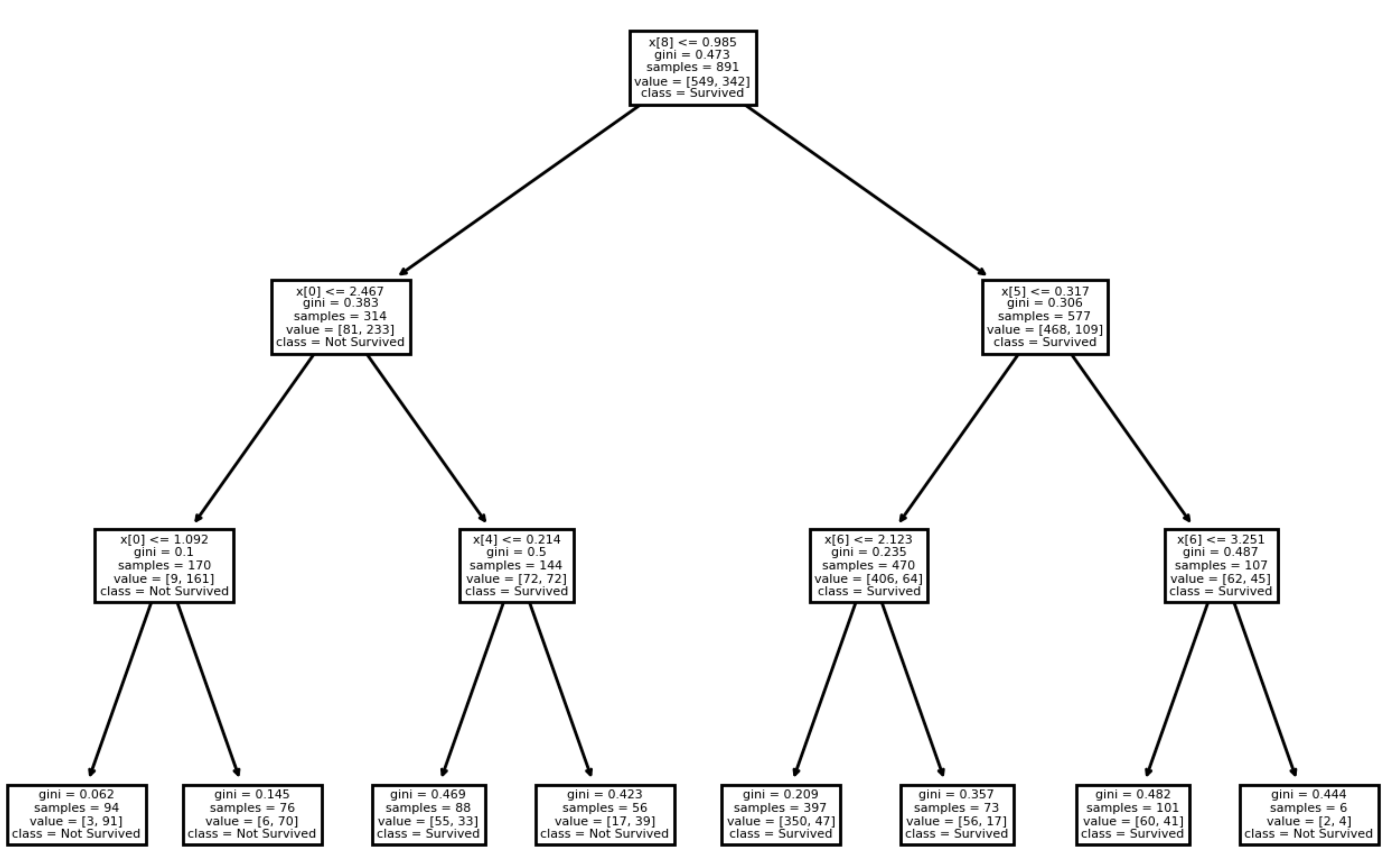
Features x[0] (Pclass/social class) and x[5] (has cabin) used to split at depth = 2

b)



The features do change. Before, it splits on x[1] (Age) 3 times and x[3] (Fare) once. Now it splits on Age 2 times and Fare both 2 times

c)



The features changes again. Now at depth = 3, it splits on x[0] (Pclass), x[4] (Embarked), and x[6] (Family Size) 2 times.

III Code in q_4.py

```python
# part 3
def split_dataset(dataset, training_ratio, validation_ratio, test_ratio):
    validation_index = math.floor(training_ratio * dataset.shape[0])
    test_index = math.floor(training_ratio * dataset.shape[0]) + math.floor(validation_ratio * dataset.shape[0])
    training_dataset = dataset[0:validation_index, :]
    validation_dataset = dataset[validation_index:test_index, :]
    test_dataset = dataset[test_index:, :]
    return training_dataset, validation_dataset, test_dataset

splits = [
    (0.80, 0.10, 0.10),
    (0.34, 0.33, 0.33),
    (0.25, 0.25, 0.50)
]

criterions = ["gini", "entropy"]

for criteria in criterions:
    for train_ratio, valid_ratio, test_ratio in splits:
        tree = DecisionTreeClassifier(max_depth=3, random_state=123123, criterion=criteria)
        training_dataset, validation_dataset, test_dataset = split_dataset(np_dataset, train_ratio, valid_ratio, test_ratio)

        training_set, training_class = np.hsplit(training_dataset, np.array([9]))
        validation_set, validation_class = np.hsplit(validation_dataset, np.array([9]))

        tree.fit(training_set, training_class)

        accuracy_count = 0

        for i, x in enumerate(validation_set):
            if tree.predict(x.reshape(1, -1))[0] == validation_class[i]:
                accuracy_count += 1

        print(f"Accuracy score for criteria:{criteria} on splits of: {train_ratio} {valid_ratio} {test_ratio} is {accuracy_count/validation_class.shape[0]}")
```

The accuracy data for all 3 splits and 2 criterions are shown below. The accuracy is represented as:

$$\frac{\#\text{ of correct prediction}}{\text{length of test/validation set}} \times 100$$

Criterion \ Splits:	80/10/10	34/33/33	25/25/50
Gini	Validation: 84.3% Test : 78.9%	Validation: 82.7% Test : 81.7%	Validation: 81.9% Test : 81.4%
Entropy	Validation: 85.6% Test : 81.1%	Validation : 80.6% Test : 79.3%	Validation: 82.9% Test : 79.9%

It seems like when the tree is built using the Entropy criteria and the dataset having an 80/10/10 split yields the best results. It has the highest validation accuracy (lowest validation error)

IV. As the max_depth parameter changes, the number of levels of the tree increases/decreases and the number of features that the tree splits on increases/decreases. Increasing the max_depth can capture more complex relationships between features which may result in higher accuracy. However, issues such as overfitting, greater sensitivity to noise and greater computational loss may occur.