**Lab 2: Decision Trees in Python**

**What to submit:** a single word/pdf file with answers for **Questions 1, 2 and 3**.

You’ll need two files to do this exercise: **Lab2.py** (the Python script file) and **Titanic.csv**. Both of those files can be found on the Blackboard. The data file contains 714 passengers in the Titanic tragedy with demographic information and whether the passenger survived.

# Before you start:

* Download both files and save them to the same folder where you keep your Python files
* Make sure that the file names are the same as “Lab2.py” and “Titanic.csv”
* Make sure that you have internet access (in order to install additional packages from the web)

# Part 1: Look at the Data File

1. Open the **Titanic.csv** data file in Excel. (If it warns you that the file format and extension don’t match and that detects that it is a SYLK file, that’s ok. Just click “Yes” and then “OK.”)
2. You’ll see something like this:

Table

Description automatically generated with medium confidence

This is the raw data for our analysis. You can see the first variable (field) is called PassengerID, the second variable (field) is called Survived, the third variable (field) is called Pclass, and so on.

The remaining lines of the file contain the data for each passenger. So the ID for the first passenger is 804, the Survived for the first passenger is 1, the Pclass for the first passenger is 3, and so on.

This is a full list of the variables:

|  |  |
| --- | --- |
| **Variable Name** | **Variable Description** |
| PassengerID | Passenger identification number |
| Survived | Whether a passenger survived or not: 0 = No, 1 = Yes |
| Pclass | Tick class: 1 = 1st, 2 = 2nd, 3 = 3rd |
| Sex | Sex: male and female |
| Age | Age in years |
| sibsp | # of siblings / spouses aboard the Titanic |
| parch | # of parents / children aboard the Titanic |

Categorical outcome Variable: **Survived** (1 = did survive, 0 = did not survive).

We will use this dataset to predict whether a passenger survived based on any combination of the remaining variables (i.e., Pclass, Age, Sex, etc.).

Some variables, like ID, are irrelevant to the analysis.

1. Close the Titanic.csv file. If it asks you to save the file, choose “Don’t Save”.

# Part 2: Explore the Lab2.py Script

1. Open the **Lab2.py** file in PyCharm. This contains the python script that performs the decision tree analysis.

Make sure the code is “colorized” – meaning that the text color is changing depending on its use.

*If all the text is black-and-white then there is a problem with your script. Usually it means that your file has a .txt extension (i.e., dTree.r.txt).*

1. Lines 2 through 9 load the packages necessary for decision tree analysis.
2. Line 21~24, we define the variables.

Text

Description automatically generated

1. Look at line 28. These lines partition the dataset into two subsets, a training set and a testing set.

* The training set will be used to create the decision tree model;
* And the testing set will be used to evaluate the classification accuracy of the model.

1. Now let’s look at **the decision tree model**. Scroll down to line 21~24:

Text

Description automatically generated

The **DecisionTreeClassifier()** function is use to classify the data into a decision tree

* The results of the clf() function will be stored in a variable called “results”.
* The formula for a decision tree model is outcome ~ X.
* Survived is the outcome you’re trying to predict (1 = did survive, 0 = did not survive).
* All variables in X are predictor variables. **Not all of those will be important enough to be included in the decision tree.**
* criterion{“gini”, “entropy”, “log\_loss”}, default=”gini”

The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “log\_loss” and “entropy” both for the Shannon information gain,

* max\_depth: int, default=None

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

* min\_samples\_split: int or float, default=2

The minimum number of samples required to split an internal node:

If int, then consider min\_samples\_split as the minimum number.

If float, then min\_samples\_split is a fraction and ceil(min\_samples\_split \* n\_samples) are the minimum number of samples for each split.

* You can use different values for max\_depth and min-samples\_split if you want a different tree.

1. Line 39~41 visualize the tree results.

Text

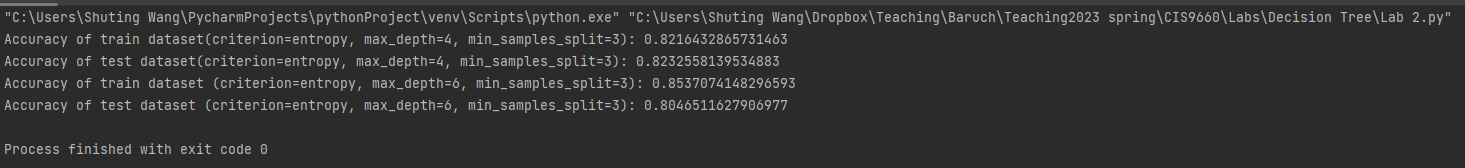
Description automatically generated

1. Line 46~47 calculate the accuracy rate for the training and testing set

A screenshot of a computer

Description automatically generated with medium confidence

# Part 3: Execute the Lab2.py Script

1. Run the script. Select Run /Run / Lab2. You’ll see a lot of action in the Console window at the bottom side of the screen, ending with this:  
     
   
2. And you’ll see the decision tree results in your working folder:

Diagram, engineering drawing

Description automatically generated

**(Tree #1)**

(It may be difficult to read the decision tree plot – the text is quite small. Open the folder of your working You should see a file called **treeplot.png**. Open the file and you will see the tree plot.)

# Part 4: Read the Decision Tree

The tree has **14** *leaf nodes* (nodes with nothing beneath them). Each of those 14 leaf nodes represents a prediction based on a combination of predictor variables.

**Correct classification rate** (see the results in the Console window)

The tree correctly predicts whether a passenger would survive 82.16% of the time for the training set, and 82.33% of the time for the validation set.

**Predictor variables**

Male is the best predictor, as it is the first split. It creates the most differentiation (separation) between survivals and non-survivals. If the answer to the question “Male<=0.5” is True, go left; if the answer is False, go right.

Pclass and Fare are the next best predictors.

Check the branch on the left: for those female passengers with Pclass lower than or equal to 2.5, then Fare.

**Samples**

The number of observations in the node.

**Value**

The value line in each box is telling you how many samples at that node fall into each category, in order. That's why, in each box, the numbers in value add up to the number shown in sample. For instance, in the first box, 305+194=499. So this means if you reach this node, there were 305 data points in category 0 (not survived), and 194 data points in category 1 (survived).

**You can read the left branch of the tree like this:**

*If you’re female and have a ticket class of 2.5 or lower then you’ll survive Titanic only 8.65% (=95/(9+95)) of the time.*

*Females who have a ticket class of 2.5 or lower and pay fare less than 28.856 will survive 83.33% (=35/(7+35))of the time.*

*Males who paid the ticket fare less than 15.645 will survive 10% (=19/(171+19)) of the time.*

Note: Variables can appear twice within a branch of the tree if it further differentiates between outcomes -- Note that Age appears multiple times in the right branch.

**Question 1:**

Based on the tree you’ve generated, how likely is it that the following passengers will survive:

A Male with a ticket fare of 15?   
B 30 year-old male who pays 35 for a ticket of class 1? C 30 year-old female with a ticket of fare 20 and ticket class of 3?

D 23 year-old female who pays 15 for the ticket of class 4?

**Will all predictor variables appear in the decision tree?**

Notice that some predictors we have included in the function may not appear in the decision tree. This is because the decision tree algorithm determined that they didn’t contribute enough to meaningfully differentiate between survivals and non-survivals.

# Part 5: Change the Parameters

## What if you use a larger max\_depth?

1. *Go to line 27 and change max\_depth from 4 to 6.*
2. Run the script by selecting Run/Run /Lab 2. You’ll now see this decision tree.

Diagram, engineering drawing

Description automatically generated

**(Tree #2)**

We can clearly see the tree is more complex (it has 20 leaf nodes now, instead of 14). That is, **larger max\_depth → more complex tree.**

**Question 2:**

Which tree has higher accuracy rate, Tree #1 or #2?

Does increasing max-depth always lead to higher accuracy rate of the testing set?

## Now, what if you use a larger min\_samples\_split?

1. *Go to line 33 and change min\_samples\_split from 3 to 20 (keep the max\_depth as 4).* That means that the tree will be less complex (less nodes) because it requires a greater number of minimum observations in each node.
2. Run the script by selecting Run/Run /Lab 2. You’ll now see this decision tree.

Diagram, engineering drawing

Description automatically generated

**(Tree #3)**

This new tree has 10 leaf nodes, as opposed to our first tree with 14 nodes and our second tree with 20 nodes. Increasing the min\_samples\_split factor threshold makes the tree simpler, and also a little less accurate (81.76% versus 82.16% of the previous trees for the training set).

That is, **larger MINSPLIT → less complex tree & lower accuracy in the training set.**

# Part 7: Which tree do we use?

We can adjust the complexity factor to find the best tree for our analysis, balancing complexity (too many nodes make the results difficult to read and interpret) and accuracy (generally, more nodes create more accurate predictions and increase the correct classification rate).

We’ve generated three decision trees in this exercise, all based on the same data:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tree #** | **On page** | **max\_depth** | **min\_samples\_split** | **# of Nodes** | **Correct Classification Rate**  **(Training Set)** | **Correct Classification Rate**  **(Validation Set)** |
| 1 | 3 | 4 | 3 | 14 | 82.16% | 82.33%(best) |
| 2 | 5 | 6 | 3 | 20 | 85.37% (best) | 80.93% (worst) |
| 3 | 6 | 4 | 20 | 10 | 81.76% (worst) | 81.40% |

* We can see that Tree #2 has the highest correct classification rate in the training set
* We can also see that Tree #3 has the lowest correct classification rate in the training set
* Trees #1 is somewhere in-between.

So which tree is the best? It depends on what your goals are. If you’re trying to choose a simple tree that is “good enough,” then you’d most likely select Tree #1. If it is important to maximize decision accuracy for the testing set, in this case Tree #1 is still the best.

Note that even comparing the best and the worst trees, there does not seem to be that much difference between the best and the worst. However, consider scale: If a grocery chain has 500,000 customers per year, the ability to improve your decision accuracy by 2% means you can identify 10,000 potential buyers that your competitors may have missed.

**Question 3:**

Based on the analyses we have done so far, use your own words to summarize how **max\_depth** and **min\_samples\_split** can alter the decision tree.