TDT4171 Assignment 4

Henrik Øberg Myhre

March 17, 2021

Exercise 1

Among the available columns, I have chosen to exclude the following:

- Name
- Ticket
- Embarked

Name could potentially be a suitable predictor with enough data. E.g., people with foreign names may not understand the English voice messages. However, with this amount of data, I will not include name.

Ticket does not provide any valuable information that is not included in fare or cabin. It may not have any valuable information at all. I cannot find a pattern in the ticket numbers. The only valuable information it may have is the placement of the cabin, which is provided in cabin, and perhaps fare. Therefore, ticket is also excluded.

Embarking should not affect any valuable information. It may decide where they sleep, but this information is stored in cabin. Hence, embarking is excluded. It probably does not decide where they sleep anyway. I assume that the people boarding on all the different places are in average very similar. E.g. that there are no reasons for some being worse swimmers than others based on Embarking.

a)

These are the remaining discrete columns:

- Sex
- Pclass

These are the remaining continuous columns:

- Age
- SibSp
- Parch
- Fare
- Cabin

These columns are continuous for the following reasons:

- Age and Fare can be any positive number, including decimal numbers.
- SibSp and Parch can also be many positive number, which makes them difficult to use discretely.
- Cabin consists of unique strings, which is also considered continuous.

This is the output for task 1a:

```
{'Sex': {'female': {'Pclass': {1: 1, 2: 1, 3: 1}},
         'male': {'Pclass': {1: 0, 2: 0, 3: 0}}}}
0.8752997601918465
```

The tree is at the end of the document. Both the full tree and a simplified version.

The accuracy of the model is, as printed, 0.8753.

b)

Including continuous variables most often increased the accuracy. If I maximize the number of possible splits, the accuracy is 0.8801. (The splitting method can be seen in the find_attribute_split_values-function.) Four splits resulted in the highest accuracy I could find, which was 0.8873. I think that this is more or less random. With enough data, more splits will probably result in a higher accuracy. I will be using the max number of splits to create the tree, even though this accuracy is less than the accuracy from four splits.

This is the tree and accuracy output for b:

```
Task 1b
{'Sex': {'female': {'Pclass': {1: {'Parch': {'<= 1.8': {'Fare': {'<= 28.96': {'SibSp': {'<= 0.0': 1,
                                                                                       '> 0.0': 1}},
                                                                  '> 28.96': 1}},
                                                '> 1.8': {'Fare': {'<= 151.8': {'SibSp': {'<= 0.0': 1,
                                                                                          '> 0.0': 1}},
                                                                   '> 151.8': 1}}}},
                                    2: {'Fare': {'<= 21.97': {'Parch': {'<= 0.0': {'SibSp': {'<= 0.0': 1,
                                                                                             '> 0.0': 1}},
                                                                        '> 0.0': 1}},
                                                 '> 21.97': 1}},
                                    3: {'Fare': {'<= 23.97': {'Parch': {'<= 3.6': {'SibSp': {'<= 0.0': 1,
                                                                                             '> 0.0': 1}},
                                                                        '> 3.6': 0}},
                                                 '> 23.97': {'Parch': {'<= 0.0': 1,
                                                                       '> 0.0': {'SibSp': {'<= 5.33': 0,
                                                                                           '> 5.33': 0}}}}}}}}},
        'male': {'Pclass': {1: {'Fare': {'<= 151.8': {'Parch': {'<= 1.8': {'SibSp': {'<= 0.0': 0,
                                                                                     '> 0.0': 0}},
                                                                '> 1.8': 1}},
                                         '> 151.8': {'SibSp': {'<= 0.0': {'Parch': {'<= 0.0': 0,
                                                                                    '> 0.0': 0}},
                                                               '> 0.0': 0}}}},
                            2: {'Parch': {'<= 0.0': {'Fare': {'<= 13.98': {'SibSp': {'<= 0.0': 0,
                                                                                     '> 0.0': 0}},
                                                              '> 13.98': 0}},
                                          '> 0.0': {'Fare': {'<= 33.96': {'SibSp': {'<= 0.0': 0,
                                                                                    '> 0.0': 0}},
                                                             '> 33.96': {'SibSp': {'<= 0.0': 1,
                                                                                   '> 0.0': 0}}}}}},
                            3: {'Fare': {'<= 46.94': {'Parch': {'<= 0.0': {'SibSp': {'<= 0.0': 0,
                                                                                     '> 0.0': 0}},
                                                                '> 0.0': {'SibSp': {'<= 1.78': 0,
                                                                                    '> 1.78': 0}}}},
                                         '> 46.94': {'Parch': {'<= 0.0': {'SibSp': {'<= 0.0': 1,
                                                                                    '> 0.0': 0}},
                                                               '> 0.0': 0}}}}}}}}
Accuracy: 0.8800959232613909
```

The tree is simplified in a tree graph at the end of the document.

c)

I originally expected the accuracy in task 1a to be lower than 87%. The high accuracy proves the importance of sex for survival. I printed the temporary p and n counts in the code, and saw that ticket class had an effect, but not enough to shift the majority of the outcomes. Given that Sex is very decisive, I expected closer to 90% accuracy in task 1b. The highest result I got from 1b was close to 89%, so this guess was quite accurate. I.e., the performance difference was as expected. If age would have been included, I think the accuracy would have gone above 90%, because I think that Sex and Age are the most important factors. As shown in 1a, Sex alone has an accuracy of approximately 87%.

Improvement suggestions

- Currently, the split-values for attributes with continuous values works optimally for values with even spacing. E.g. the values [0, 1, 2, 3, 4, 5, 6] will create split-values similar to: [0, 0.9, 2.8, 3.7, 4.6, 5.5, 4.3, 5.2]. Since the operators are > and <=, every possible splitting-scenario will be tested in this case. However, if the values are not evenly spaced, e.g.: [0, 0.5, 0.6, 7], the resulting split-values will be something like: [0, 2.5, 5.3], which does not include a value between 0.5 and 0.6. This may be the case for some fare-values. Therefore, the algorithm could be improved by iterating through all unique values and creating a split-value between each training-value. This would also exclude multiple split-values between the same input-values. This solution would increase the processing time.
- As shown by the output dictionary-trees in 1a and 1b, they have some unnecessary long branches. These are removed in the simplified versions. The long branches will decrease the testing speed because the test will have to go down the irrelevant sub-tree. If all the branches in sub-tree produces the same value, the sub-tree can be replaced with that value before testing. E.g. the tree in 1a can be replaced with its simplifies version, which only includes Sex.
- Another improvement could be pruning by information gain. Pruning attributes that apparently are not relevant could increase the processing speed, and remove unnecessary noise. The processing speed would be increased because there would be less nodes to visit for the testing-data. Chi-squared pruning is one specific example.

Exercise 2

There are two columns with missing values in the provided data:

- Age
- Cabin

Cabin has very few values, and is therefore difficult to use for testing. Age has few missing values, and is also likely very important for the prediction, because of "women and children first"-policy.

Ignoring

While creating the decision-tree, we can ignore the rows that has columns with no values. This will reduce the size of the training data, but it may improve the tree if the attribute with missing values is important enough. I guess that including Age would improve the decision tree and the accuracy, while including Cabin would do more harm than good by reducing the size of the training data.

When testing, N/A can be solved by e.g. guessing when the algorithm encounters a missing value. The algorithm could also make an educated guess by e.g. examining what value is most frequent among the children of the attribute with a missing value. This could return the correct prediction a majority of times, but it will reduce the accuracy. We could also ignore the testing-rows with no-value-columns, but this would obviously be worse since it will never predict rows with no values correct.

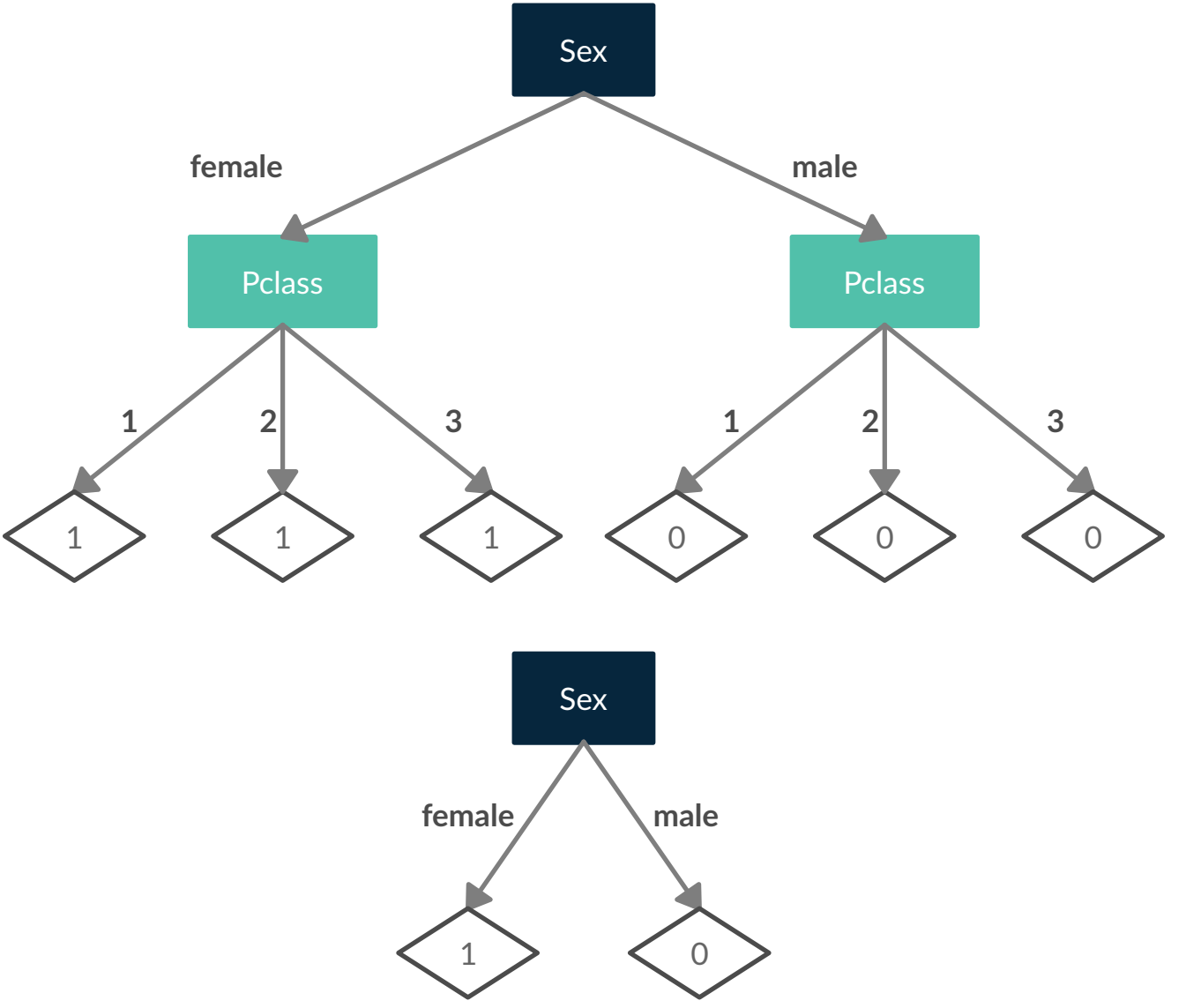
Median

Another possible solution is to assign the median value for the specific column to the N/A-values. This requires the values to be numbers, and not strings or such. It would work for Age, but not for Cabin. In addition to not being a universal solution, this could produce patterns that we would normally not see, which may negatively affect the accuracy. We do not miss any data by using the median, which is positive.

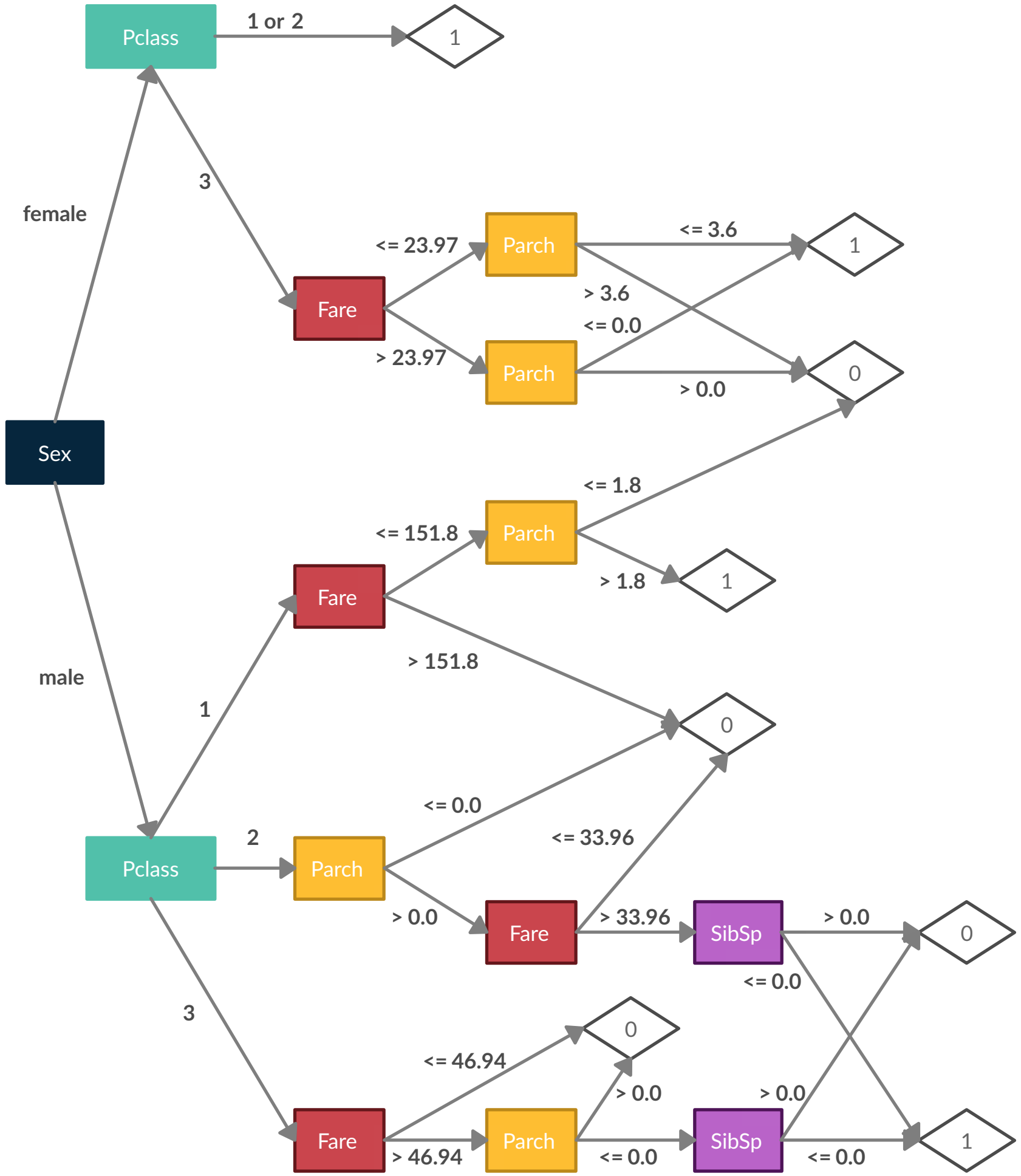
N/A-node

Instead of assigning a fake value to the column with no value, the algorithm can create a new node named N/A. This can be done easily with discrete attributes by adding N/A or whatever the non-existing value is saved as. Continuous attributes would require the algorithm to implement checks in both training and testing. Could potentially check if none => save as N/A (or in testing: access tree[N/A]). This is probably the best option for Cabin, since the two other options have to many negative effects, as explained.

Task 1a


Sex
female
male
Pclass
Pclass
1
2
3
1
2
3
1
1
1
0
0
0
Sex
female
male
1
0

Task 1b


Pclass
1 or 2
1
3
female
Sex
Fare
<= 23.97
Parch
<= 3.6
1
> 3.6
<= 0.0
> 23.97
Parch
0
> 0.0
<= 1.8
Parch
<= 151.8
> 1.8
1
Fare
male
> 151.8
0
1
<= 0.0
2
Pclass
Parch
<= 33.96
> 0.0
Fare
> 33.96
SibSp
> 0.0
0
<= 0.0
3
0
<= 46.94
> 0.0
> 0.0
Fare
Parch
SibSp
1
> 46.94
<= 0.0
<= 0.0