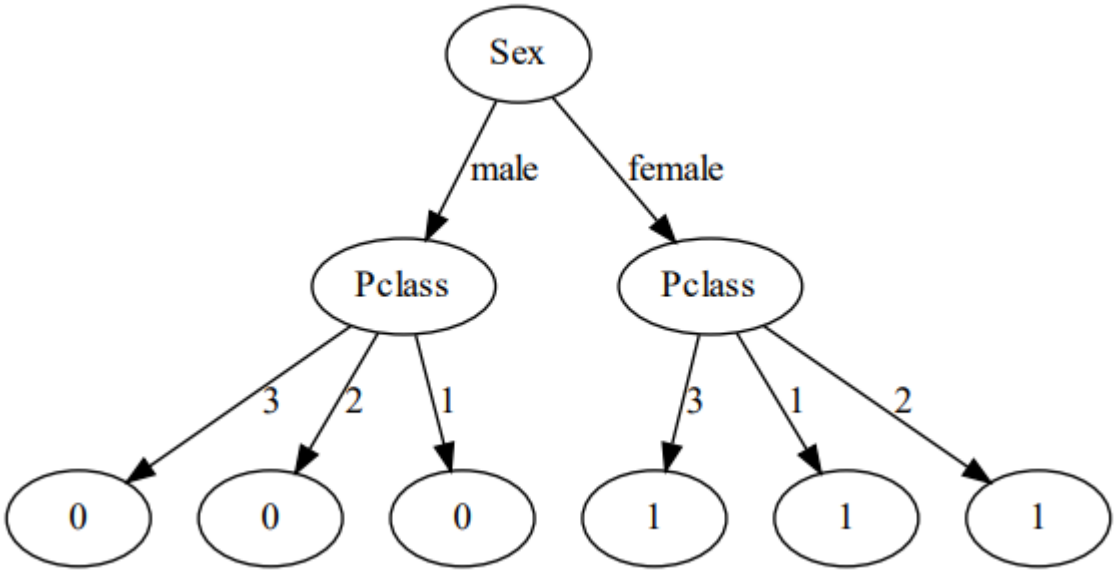
Assignment 4 – Decision Trees

1 – Decision Trees



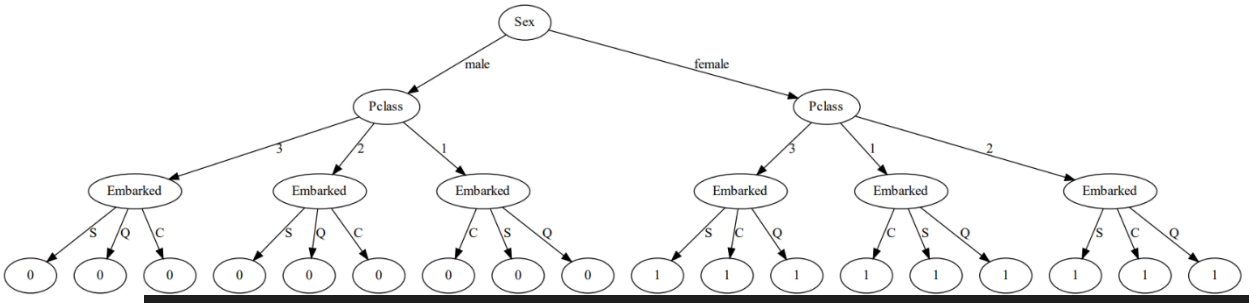

```
haako@DESKTOP-CLSCP44 MINGW64 /c/prog/tdt4171/titanic (master)
$ py decisionTree.py
Accuracy of decision network:  0.8752997601918465
```

a) This is the decision tree of the Titanic training dataset with the relevant attributes.
We were supposed to only care for categorical variables and drop variables where data was missing.
The columns which are continuous, and thereafter not used, are:

- Age – are continuous and at the same time have missing data.
- SibSp – can be considered as both categorical and continuous, but we assume it is more like a continuous because it can have 7 different values between 0 and 8. The node for this would therefore have 7 branches which is too much, therefor continuous. Would work though with a threshold and categorize in 2 or maybe 3 values from these 7.
- Parch – The same argumentation as SibSp, 8 values between 0 and 9.

Other values that does not help us when trying to make decisions are name, ticket, fare and cabin number. Embarked could maybe help us a bit, but here we assume that the probability of surviving does not depend on where you embarked the ship. Cabin number could help us, but it is missing a lot of data and we do not know how the cabins are located in the ship considered their name.

Name have no discriminating property at all, it makes no sense to include this in the tree. At the same time, we assume ticket, fare and cabin number does not help us either, it can be that it gives a small percentage improvement on training set. Most likely won't and will increase complexity of resulting decision tree. Anyway, we assume that this has a low probability, and is more damaging then improving when we think of higher computation time.




```
haako@DESKTOP-CLSCP44 MINGW64 /c/prog/tdt4171/titanic (master)
$ py decisionTree.py
Accuracy of decision network:  0.8752997601918465
```

One could as said consider embarked as a discrete variable as well. If we implement it, we can see the tree structure got bigger, but the accuracy is still the same. This means it does not provide any more useful information, so we could easily just remove it (pruning) and use the first decision tree.

b) Did not finished this part… Think I had a good start, and it is commented into the code as pseudo what I would do and where I would do it. Anyways, I needed answers for this to do c), but lucky me heard some rumors that the accuracy was similar. I will therefore assume that the tree got bigger but had no higher precision.

c) I did not manage to create the best split which would been in the information gain, see TODO in code and comments. Anyway, I heard some rumors that the performance is almost equal. This is a bit surprising because we could imagine knowing if someone have family members could help us predict if they survived or not given that mothers with children were saved while single women were not, but it is not the case here. It can be that the sex and ticket class is dominating, so the effects from them continuous variables (family members) does not manipulate the decisions. Since the first tree with only pClass and sex had the same accuracy as all the bigger trees, we could just choose to use that instead of bigger trees with no more gain. This would improve the execution time when deciding the outcomes from the test set.

Another method to implement could be to handle variables that could make a great impact on the accuracy (positively), even if it misses values for some sets. How to handle this is mentioned in part 2, but this would for sure improve our accuracy, especially age.

Summed up, two methods that could improve our performance is to prune the tree by removing nodes/variables that does not change the outcome or implement methods to handle sets that sometimes misses values. The first would make our tree less complex and have a faster execution time. The second method could make it more precise and give it higher accuracy, given that the data is "important" and makes a difference. And not to many missing.

2 – Missing Values

One of the columns that with great probability would improve our performance is the age. Since, as we know, children entered the emergency boats first, people with a low age would have a much greater probability of surviving. This could make our decision tree even better and more precise.

Two methods to handle missing data:

- One method could be to act as all the data were present and handle it when we are trying to read values not present. We have two options when encountering a missing value.
  The first thing we could do is to handle it as the third elif sentence in the dtl-algorithm. When we discover the missing value, we consider the plurality value for the parent. This can be a good way if the node is deep in the tree and the entropy has been reduced a lot from the top. The dominant value of the parent can therefore be a good assumption. If the node for the missing value are far up in the tree, this could be less efficient, even random. That gives us the second method.

- If we meet upon missing data, we can create a clone where both values are true, searching further down in both subtrees, and if both decisions are the same, we can make that decision. This means the value with missing data would not affect the outcome anyway, and this is for sure a better method than the plurality value of the parents. If the decisions are unequal, we could make the decision random, or our third method; combining it with the plurality value.

Summed up, we can consider the plurality value of the parent, or create a clone and discover the decision for the value being both true and false and combining them. The best method is maybe to combine these, first creating a clone and discovering the leaves, and if unequal consider plurality value of parent, or randomly.

Afterall, it depends on the amount of data missing if these are efficient and worth implementing. If there are only a few rows missing data, the methods will probably work well because of the greater amount of data (supposing it gives us better accuracy). If a lot of the data is missing, the second method and combining them would for sure make the creation slower. It could also make it worse, if it makes random choice when it could maybe find clean decisions if we let out the node.

Another method could be to just remove the rows with missing data, or even the columns. The latter is the method we used in the task, but I do suspect this decreased the accuracy a lot, especially removing the column for age.

Other methods I can think of, which will not however be describe in detail, are to use the average or median value of that variable from the whole dataset with the value present. This will most likely work best on continuous variables though, for discrete this will be the same as using max-value, the value for the variable that is most used.

I am sorry for my English, started a bit late and wrote this in a wind, as we say in Norwegian:)