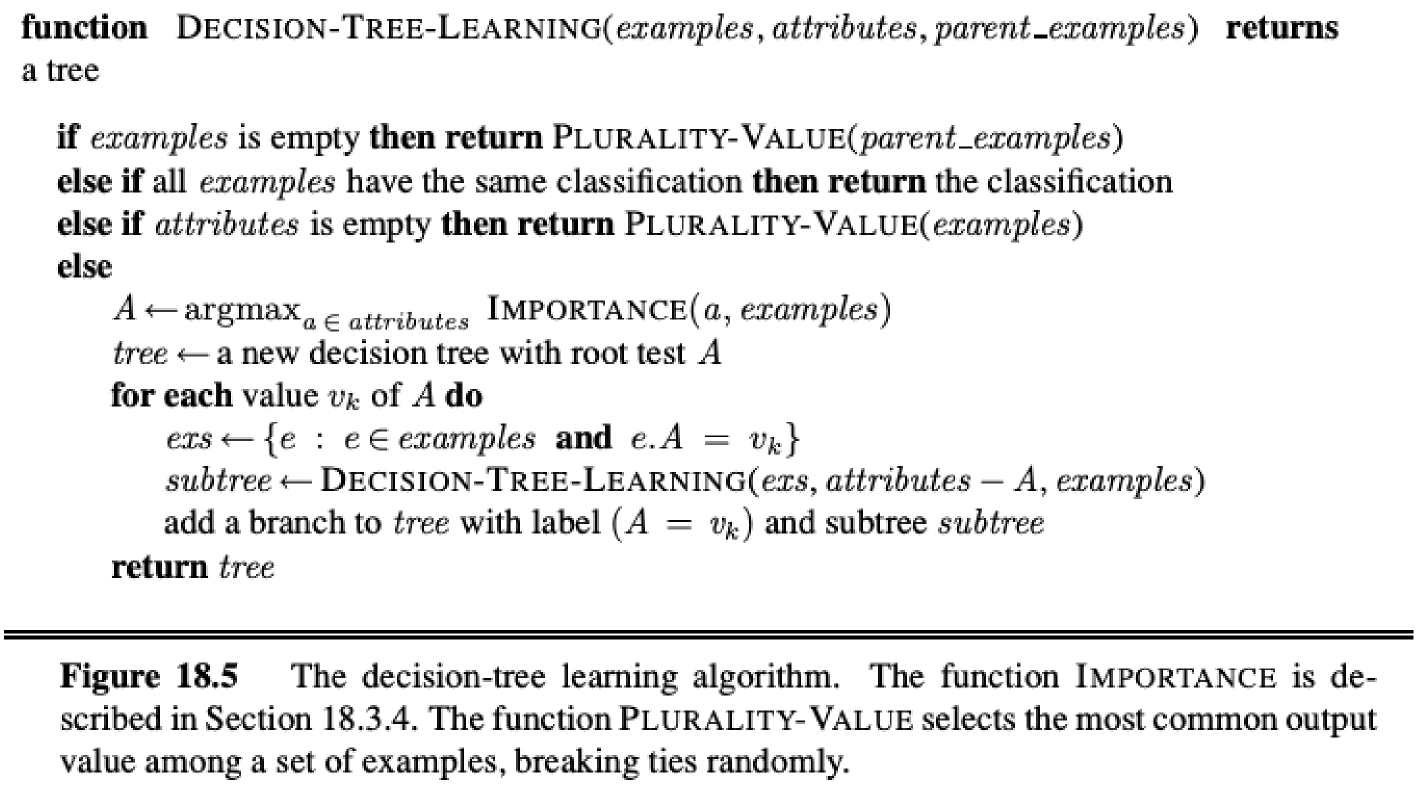
TDT4171 - Assignment 4

Hanna Waage Hjelmeland, MTTK

Task 1

1a)

We are to implement the following algorithm that can support categorical variables:



```
function DECISION-TREE-LEARNING($examples$, $attributes$, $parent\_examples$) returns a tree
  if $examples$ is empty then return PLURALITY-VALUE($parent\_examples$)
  else if all $examples$ have the same classification then return the classification
  else if $attributes$ is empty then return PLURALITY-VALUE($examples$)
  else
     $A \leftarrow \mathrm{argmax}_{a \in attributes}$ IMPORTANCE($a$, $examples$)
     $tree \leftarrow$ a new decision tree with root test $A$
     for each value $v_k$ of $A$ do
        $exs \leftarrow \{e : e \in examples$ and $e.A = v_k\}$
        $subtree \leftarrow$ DECISION-TREE-LEARNING($exs$, $attributes - A$, $examples$)
        add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
     return $tree$
```

Figure 18.5 The decision-tree learning algorithm. The function IMPORTANCE is described in Section 18.3.4. The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

Where the IMPORTANCE-function consists of the information-gain of the attribute A, which is the "expected reduction in entropy" (AIMA, p 704). The information-gain is found by:

$$Gain(A) = B(\frac{p}{p+n}) - Remainder(A)$$

where

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k+n_k}{p+n} B(\frac{p_k}{p_k+n_k})$$

and

$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$$

(retrieved from AIMA, p 704).

The IMPORTANCE function will return the attribute with the maximum gain, which the algorithm will choose as the root. This minimizes the remaining information needed to classify new examples (slides lecture 7, p 25).

We take a look at the training data: (10 first rows of train)

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	0	3	Hagland, Mr. Konrad Mathias Reiersen	male	NaN	1	0	65304	19.9667	NaN
1	1	3	Aks, Mrs. Sam (Leah Rosen)	female	18.0	0	1	392091	9.3500	NaN
2	1	1	Young, Miss. Marie Grice	female	36.0	0	0	PC 17760	135.6333	C32
3	0	2	Gale, Mr. Shadrach	male	34.0	1	0	28664	21.0000	NaN

And the test data:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	1	Schabert, Mrs. Paul (Emma Mock)	female	35.00	1	0	13236	57.7500	C28
1	1	3	Jansson, Mr. Carl Olof	male	21.00	0	0	350034	7.7958	NaN
2	1	2	Corey, Mrs. Percy C (Mary Phyllis Elizabeth Mi...	female	NaN	0	0	F.C.C. 13534	21.0000	NaN

I chose to disregard the following attributes:

- PassengerID (I assume this is approximately random and has little connection to the attributes of the passenger. It will also make the tree unnecessarily complex and cause for the tree to memorize the passengers in the training set rather than generalizing)
- Name (except for maybe a few "famous" last names, I think this will have little effect on the predictions. Also, this makes the tree prone to memorization.)
- Ticket number (same argument at passenger ID)
- Cabin (lacks values for most of the passengers, and I assume the characteristics that can be found here is picked up by e.g. passenger class)
- Age (lacks values)

Disregarded attributes due to the attribute being continous:

- Passenger fare

Disregarded attributes due to testing and not seeing statistically significant effects:

- SibSp
- Parch

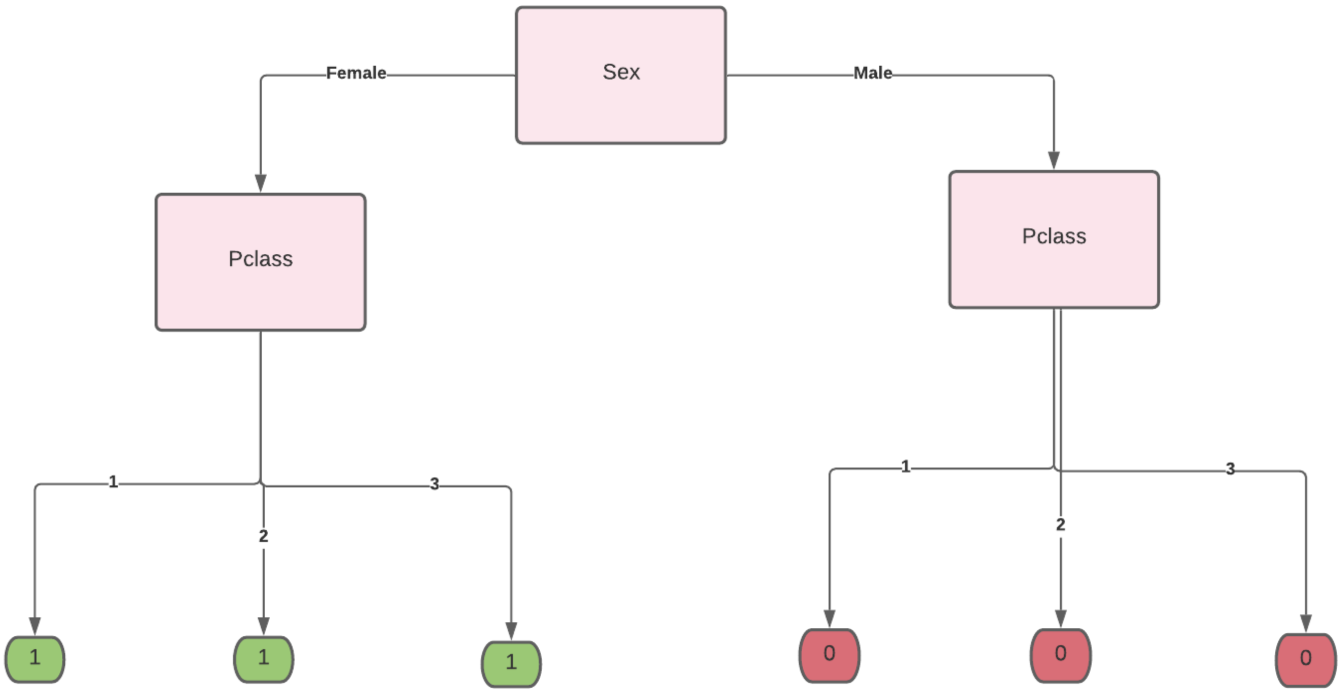
We are then left with: (showing 10 first rows of train set)

	Survived	Pclass	Sex
0	0	3	male
1	1	3	female
2	1	1	female
3	0	2	male
4	0	3	male
5	0	3	male

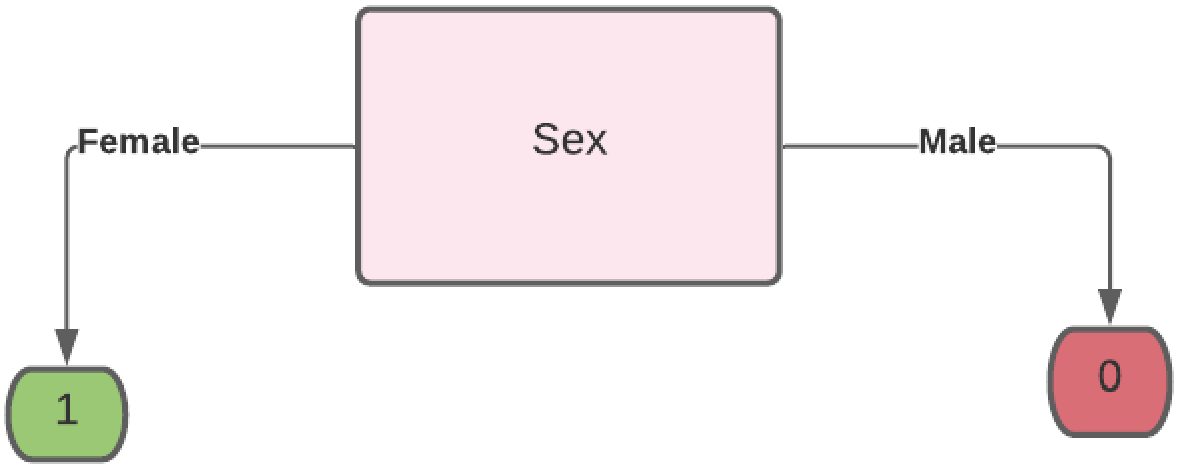
Resulting of building decision tree:

```
Sex



branch:  female , from  Sex
Pclass


branch: 1 val: 1
branch: 2 val: 1
branch: 3 val: 1


branch:  male , from  Sex
Pclass


branch: 1 val: 0
branch: 2 val: 0
branch: 3 val: 0
```

This gives the following decision tree:



Which simplifies to



(all visual representations of decision trees in this assignment are made with http://lucidchart.com)

Testing the decision tree on the test set:

```
Accuracy of model when including attributes Sex & Pclass:  0.8752997601918465
```

Out of curiosity, we check the accuracy when including the SibSp and Parch attributes as well: (I do not print the tree as it becomes quite long)

```
Accuracy of model when including attributes Sex, Pclass, Parch & SibSp:  0.87050
```

We see that the accuracy actually decreases when including these features, which can indicate that these features had little to do with the survival of the passengers. Also, this again makes the tree more prone to memorizing the passengers rather than generaliza.

1b) Continous attributes

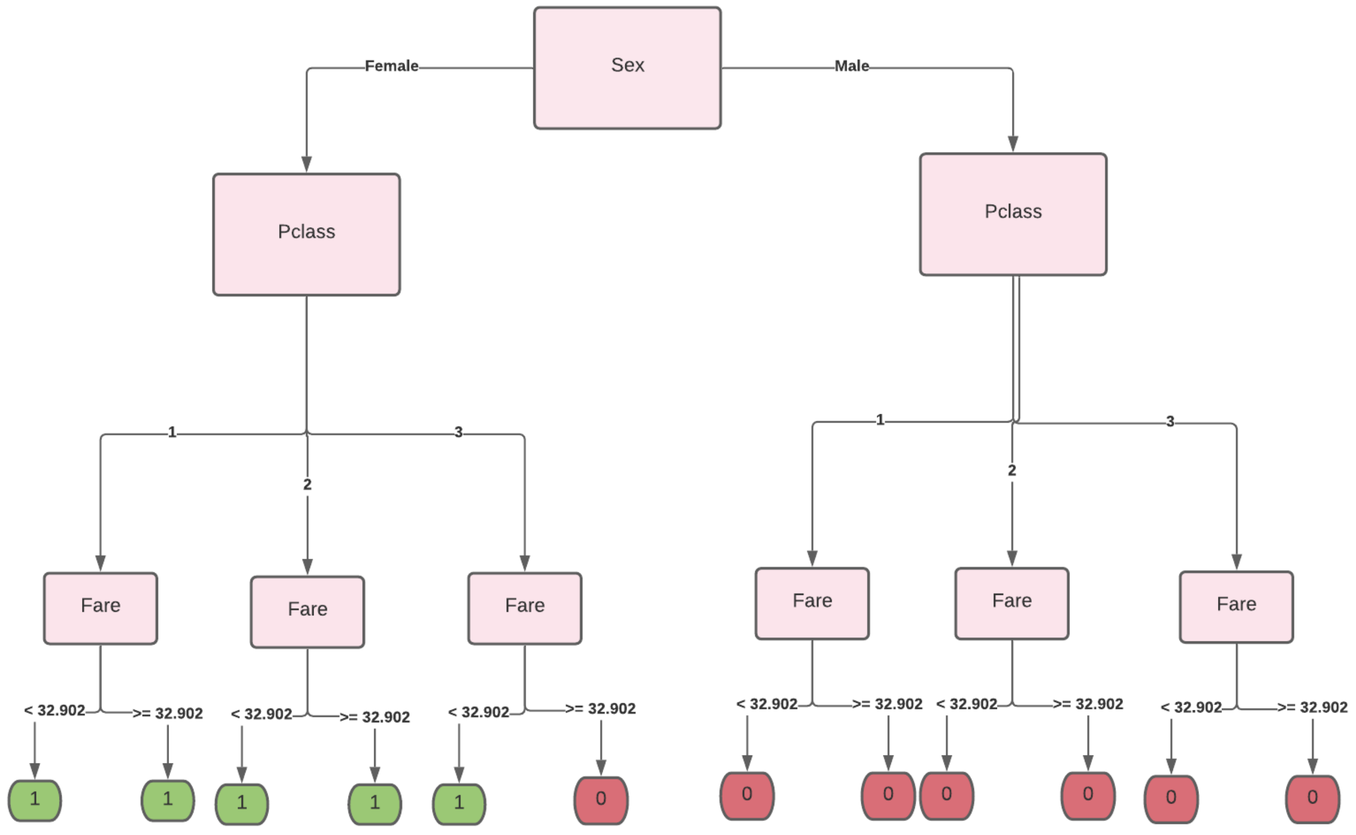
We want to include the continous variable passenger fare. We use the mean of the example data to create a split, once.

Result:

```
Sex



branch:  female , from  Sex
Pclass



branch:  1 , from  Pclass
Fare


branch: 32.902 val: 1
branch:  >= 32.902 val: 1
```

```
branch:  2 , from  Pclass
Fare


branch: 32.902 val: 1
branch:  >= 32.902 val: 1


branch:  3 , from  Pclass
Fare


branch: 32.902 val: 1
branch:  >= 32.902 val: 0


branch:  male , from  Sex
Pclass




branch:  1 , from  Pclass
Fare


branch: 32.902 val: 0
branch:  >= 32.902 val: 0


branch:  2 , from  Pclass
Fare


branch: 32.902 val: 0
branch:  >= 32.902 val: 0


branch:  3 , from  Pclass
Fare
```

This gives the following decision tree:



Testing the model:

```
Accuracy of model when including attributes Sex , Pclass & the continous variabl
```

1c)

We see that the accuracies are identical in a) & b). This was surprising, as we got a higher resolution along the branch Sex -> PClass -> Pclass(3) -> Fare in c. But this indicates that the mean we found in the start, 32.902, is not fit to split the women of class 3, as none of these belongs to the group that paid over 32.902. Thus, we get the same result when we classify. When running the code, we see that the default-value that we give to DTL is 0, which is the reason that the node value becomes 0.

To improve the classifier we could have

- implemented several splits on the continous attribute in c) - this allows for adaptive splits, which could give higher resolution

- used a validation set while training to avoid overfitting on the training set. This can be done e.g. by stopping the training when the validation accuracy stops improving - this can help us to avoid overfitting the training data.

Out of curiosity, I implemented several splits. This gave the following tree:

```
Sex



branch:  female , from  Sex
Pclass



branch:  1 , from  Pclass
Fare


branch: 110.442 val: 1
branch:  >= 110.442 val: 1


branch:  2 , from  Pclass
Fare


branch: 22.123 val: 1
branch:  >= 22.123 val: 1


branch:  3 , from  Pclass
Fare


branch: 15.658 val: 1
branch:  >= 15.658 val: 1


branch:  male , from  Sex
Pclass
```

```
branch:  1 , from  Pclass
Fare


branch: 68.922 val: 0
branch:  >= 68.922 val: 0


branch:  2 , from  Pclass
Fare


branch: 18.882 val: 0
branch:  >= 18.882 val: 0


branch:  3 , from  Pclass
```

Now, we see that we split on 15.658 instead, which gave the same tree as in task a) and thus the same accuracy:

```
Accuracy when allowing for adaptive splitting on the continous variable:  0.8752
```

Task 2

Methods proposed (can be used both in training and prediction):

- Method 1: Drop the rows that have missing data. Pseudocode on the trainingset: train = train.dropna()
- Method 2: Fill in the missing data with the mean or median of the examples. Pseudocode on the trainingset: train["Age"] = train["Age"].fillna(train["Age"].mean()/.median())

-	Method 1	Method 2
Advantages	The data is 100% authentic	We get to use all rows
Disadvatages	We miss out on rows that could have been valuable.	We construct data that arent true. This may cause the model tc

Comment on method 1: missing out on some data is not necessarily a big deal if we have a lot of data, but in the Titanic case we should be careful because the dataset is fairly small.

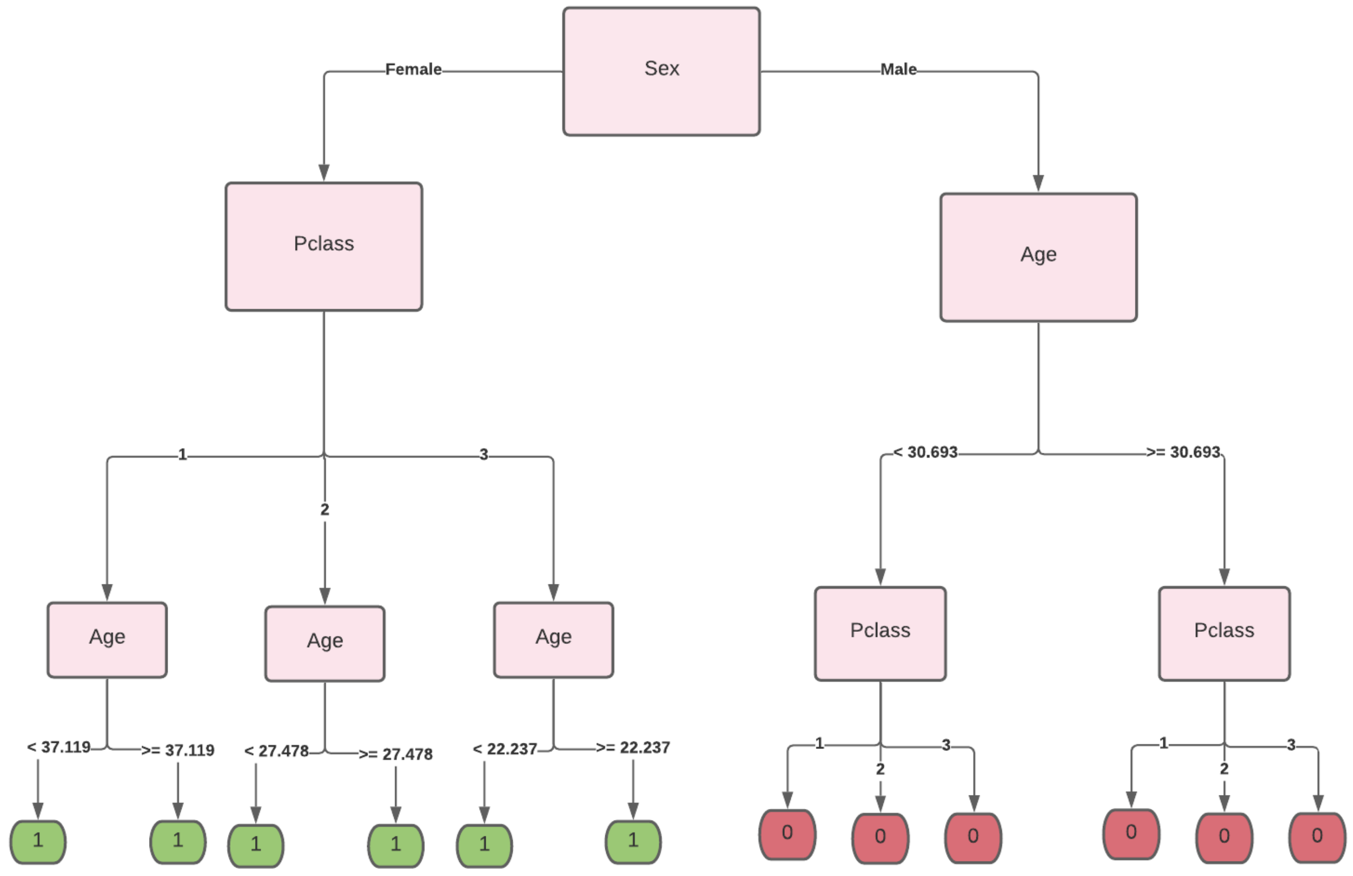
Assumptions: the data is authentic and true if it is in the dataset (not necessarily the case as this data was input manually by humans and is thus prone to human error)

Out of curiosity (hehe) I tried with the filla-method with age (together with the adaptive splitting-method from c))

```
Sex




branch:  female , from  Sex
Pclass




branch:  1 , from  Pclass
Age


branch: 37.119 val: 1
branch:  >= 37.119 val: 1


branch:  2 , from  Pclass
Age


branch: 27.478 val: 1
branch:  >= 27.478 val: 1


branch:  3 , from  Pclass
Age


branch: 22.237 val: 1
branch:  >= 22.237 val: 1


branch:  male , from  Sex
Age




branch:  30.693 , from  Age
Pclass


branch: 1 val: 0
branch: 2 val: 0
```

```
branch: 3 val: 0


branch:   >= 30.693 , from  Age
Pclass


branch: 1 val: 0
branch: 2 val: 0
branch: 3 val: 0
```

This gave the following decision tree:



which again can be simplified to the one in a). Thus, we get the same accuracy:

```
Accuracy of model when including age:  0.8752997601918465
```