Exercice 1: Algorithms

1. *Consider a single algorithm that always predicts the majority class. What would it predict, and what would be its accuracy (proportion of correct predictions)? Such a classifier quite often serves as a baseline.*
   a. **What would it predict? Mostly “yes”**
   Class: buys_computer
   Since our majority class includes more “yes” than “no”, the single algorithm will probably predict “yes”.
   b. **What would be its accuracy (proportion of correct predictions)?**
   9 well predicted

$$\frac{TP+TN}{P+N} = \frac{9+0}{5+9} = \frac{9}{14} = 0,6428 => 64,28\%$$

2. *The example “8.1 of a decision tree” using information gain calculates a multiway decision tree. Change the calculations for the attribute age to calculate a binary tree. Because this attribute is ordinal, consider only the following splits {(youth, middle-age), (senior)} and {(youth), (middle-age, senior)}. Show your calculations. Compare the values that you get for information gain with the value obtained in the book for the multi-way split. Explain the purpose of “Gain Ratio” compared to “Information Gain” in a multiway decision tree.*
   a. **Information Gain**

$$Info(D) =- \frac{9}{14}log_2(\frac{9}{14}) - \frac{5}{14}log_2(\frac{5}{14}) = 0.940\ bits$$

<u>*First case: split {(youth, middle-age), (senior)}*</u>

- (youth, middle-age) => 9 tuples in total
- 6 Yes and 3 No
- (Senior) => 5 tuples in total
- 3 Yes and 2 No

$$Info_{age}(D) = \frac{9}{14}(-\frac{6}{9}log_2(\frac{6}{9}) - \frac{3}{9}log_2(\frac{3}{9})) + \frac{5}{14}(-\frac{3}{5}log_2(\frac{3}{5}) - \frac{2}{5}log_2(\frac{2}{5}))$$
$$Info_{age}(D) = 0.7482542349\ bits$$
$$Gain(age) = Info(D) - Info_{age}(D) = 0.1917457651\ bits$$

<u>*Second case: split {(youth), (middle-age, senior)}*</u>

- (youth) => 5 tuples in total
- 2 Yes and 3 No
- (middle-age, senior) => 9 tuples in total
- 7 Yes and 2 No

$$Info_{age}(D) = \frac{5}{14}(-\frac{2}{5}log_2(\frac{2}{5}) - \frac{3}{5}log_2(\frac{3}{5})) + \frac{9}{14}(-\frac{7}{9}log_2(\frac{7}{9}) - \frac{2}{9}log_2(\frac{2}{9}))$$
$$Info_{age}(D) = 0.838042395\ bits$$
$$Gain(age) = Info(D) - Info_{age}(D) = 0.101957605\ bits$$

c. Compare the values that you get for information gain with the value obtained in the book for the multi-way split

Information Gain calculated in the book (Multiway-split)

$Gain(age) = 0.246\ bits$

Information Gain calculated for the split {(youth, middle-age), (senior)}

$Gain(age) = 0.1917457651\ bits$

Information Gain calculated for the split {(youth), (middle-age, senior)}

$Gain(age) = 0.101957605\ bits$

We can define information gain as a measure of how much information a feature provides about a class. Information gain helps to determine the order of attributes in the nodes of a decision tree.

The main node is referred to as the parent node, whereas sub-nodes are known as child nodes. We can use information gain to determine how good the splitting of nodes in a decision tree is.

It can help us determine the quality of splitting.The more the entropy removed, the greater the *information gain*. The higher the *information gain*, the better the split.

Because the *Multiway-split* has the highest *information gain*, it is selected as the best split.

d. Explain the purpose of "Gain Ratio" compared to "Information Gain" in a multiway decision tree.

Information gain:

It works fine for most cases, unless you have a few variables that have a large number of values (or classes).
Information gain is biased towards choosing attributes with a large number of values as root nodes.

Gain ratio:

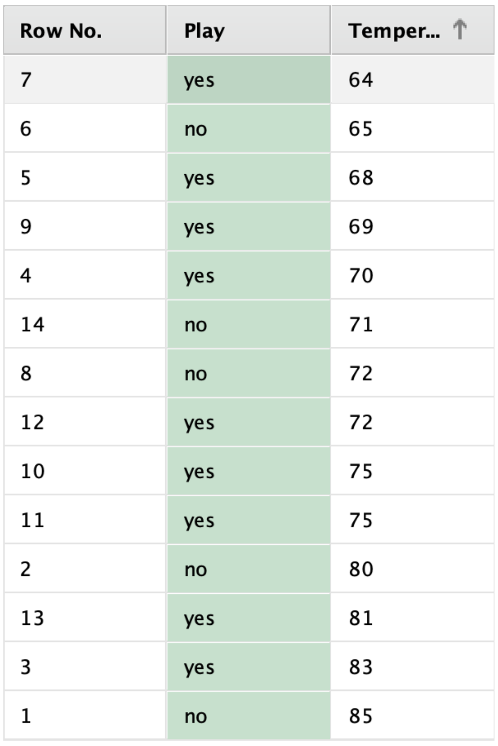
This is a modification of information gain that reduces its bias and is usually the best option. Gain ratio overcomes the problem with information gain by taking into account the number of branches that would result before making the split.
It corrects information gain by taking the intrinsic information of a split into account.

3. Consider the Golf-Dataset. The label to predict is "Play", i.e., whether one should play golf or not according to various weather-features. Explain how information gain will be calculated for the attribute Temperature; assume a binary split. Note: You do not have to do all calculations with this attribute but explain the principle.

First, we can consider the attribute *Temperature* as a *Continuous-valued attribute.* Continuous attributes can be represented as floating point variables.

To calculate the *Information-Gain* for *Continuous-Valued Attributes*, we need to start by calculating the split point.

First of all, we need to sort the data in ascending order. After sorting the data, data is shown in the table below.



Row No.	Play	Temper... ↑
7	yes	64
6	no	65
5	yes	68
9	yes	69
4	yes	70
14	no	71
8	no	72
12	yes	72
10	yes	75
11	yes	75
2	no	80
13	yes	81
3	yes	83
1	no	85

Now we calculate the midpoint between each pair of adjacent values, which can be a possible split-point.

$$midpoint = \frac{a_i + a_{i+1}}{2}$$

$\frac{64+65}{2}$	$\frac{65+68}{2}$	$\frac{68+69}{2}$	$\frac{69+70}{2}$	$\frac{70+71}{2}$	$\frac{71+72}{2}$	$\frac{72+72}{2}$	$\frac{72+75}{2}$	$\frac{75+75}{2}$	$\frac{75+80}{2}$	$\frac{80+81}{2}$	$\frac{81+83}{2}$	$\frac{83+85}{2}$
64.5	66.5	68.5	69.5	70.5	71.5	72	73.5	75	77.5	80.5	82	84

<u>Split point 64.5:</u>

$Info_{Temperature}(D) < 64.5(D)$

$= \frac{1}{14}(-\ 1log_2(1)) + \frac{13}{14}(-\frac{8}{13}log_2(\frac{8}{13}) - \frac{5}{13}log_2(\frac{5}{13})) = 0.9612366047$

<u>Split point 66.5:</u>

$Info_{Temperature}(D) < 66.5(D)$

$= \frac{2}{14}(-\frac{1}{2}log_2(\frac{1}{2}) - \frac{1}{2}log_2(\frac{1}{2})) + \frac{12}{14}(-\frac{8}{12}log_2(\frac{8}{12}) - \frac{4}{12}log_2(\frac{4}{12})) = 0.929967$

<u>Split point 68.5:</u>

$Info_{Temperature}(D) < 68.5(D)$

$= \frac{3}{14}(-\frac{2}{3}log_2(\frac{2}{3}) - \frac{1}{3}log_2(\frac{1}{3})) + \frac{11}{14}(-\frac{7}{11}log_2(\frac{7}{11}) - \frac{4}{11}log_2(\frac{4}{11})) = 0.9397964895$

Split point 69.5:
$Info_{Temperature}(D) < 69.5(D)$
$= \frac{4}{14}(-\frac{3}{4}log_2(\frac{3}{4}) - \frac{1}{4}log_2(\frac{1}{4})) + \frac{10}{14}(-\frac{6}{10}log_2(\frac{6}{10}) - \frac{4}{10}log_2(\frac{4}{10})) = 0.9253298887$

Split point 70.5:
$Info_{Temperature}(D) < 70.5(D)$
$= \frac{5}{14}(-\frac{4}{5}log_2(\frac{4}{5}) - \frac{1}{5}log_2(\frac{1}{5})) + \frac{9}{14}(-\frac{5}{9}log_2(\frac{5}{9}) - \frac{4}{9}log_2(\frac{4}{9})) = 0.8949517867$

Split point 71.5:
$Info_{Temperature}(D) < 71.5(D)$
$= \frac{6}{14}(-\frac{4}{6}log_2(\frac{4}{6}) - \frac{2}{6}log_2(\frac{2}{6})) + \frac{8}{14}(-\frac{5}{8}log_2(\frac{5}{8}) - \frac{3}{8}log_2(\frac{3}{8})) = 0.9389462163$

Split point 72: (same as Split point 71.5)

Split point 73.5:
$Info_{Temperature}(D) < 73.5(D)$
$= \frac{8}{14}(-\frac{5}{8}log_2(\frac{5}{8}) - \frac{3}{8}log_2(\frac{3}{8})) + \frac{6}{14}(-\frac{4}{6}log_2(\frac{4}{6}) - \frac{2}{6}log_2(\frac{2}{6})) = 0.9389462163$
Also same as split point 71.5

Split point 75: (same as Split point 73.5)

Split point 77.5:
$Info_{Temperature}(D) < 77.5(D)$
$= \frac{10}{14}(-\frac{7}{10}log_2(\frac{7}{10}) - \frac{3}{10}log_2(\frac{3}{10})) + \frac{4}{14}(-\frac{2}{4}log_2(\frac{2}{4}) - \frac{2}{4}log_2(\frac{2}{4})) = 0.9152077852$

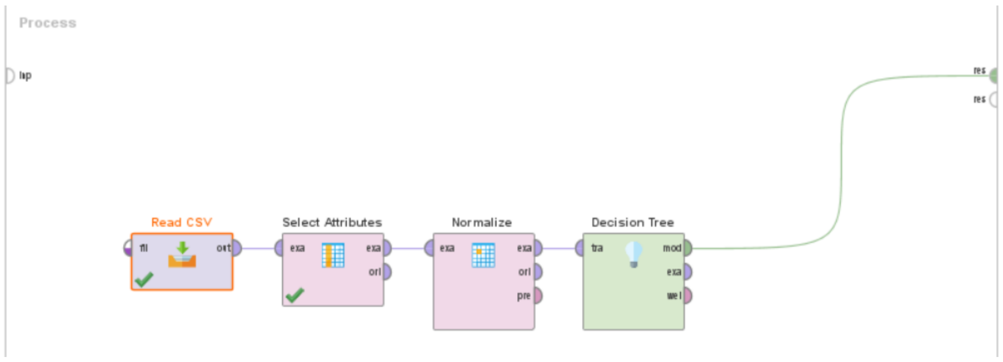
Split point 80.5:
$Info_{Temperature}(D) < 80.5(D)$
$= \frac{11}{14}(-\frac{7}{11}log_2(\frac{7}{11}) - \frac{4}{11}log_2(\frac{4}{11})) + \frac{3}{14}(-\frac{2}{3}log_2(\frac{2}{3}) - \frac{1}{3}log_2(\frac{1}{3})) = 0.9397964895$

Split point 82:
$Info_{Temperature}(D) < 82(D)$
$= \frac{12}{14}(-\frac{8}{12}log_2(\frac{8}{12}) - \frac{4}{12}log_2(\frac{4}{12})) + \frac{2}{14}(-\frac{1}{2}log_2(\frac{1}{2}) - \frac{1}{2}log_2(\frac{1}{2})) = 0.929967$

Split point 84:
$Info_{Temperature}(D) < 84(D)$
$= \frac{13}{14}(-\frac{9}{13}log_2(\frac{9}{13}) - \frac{4}{13}log_2(\frac{4}{13})) + \frac{1}{14}(-1log_2(1)) = 0.8268850945$

Exercise 2

1. Read the data.

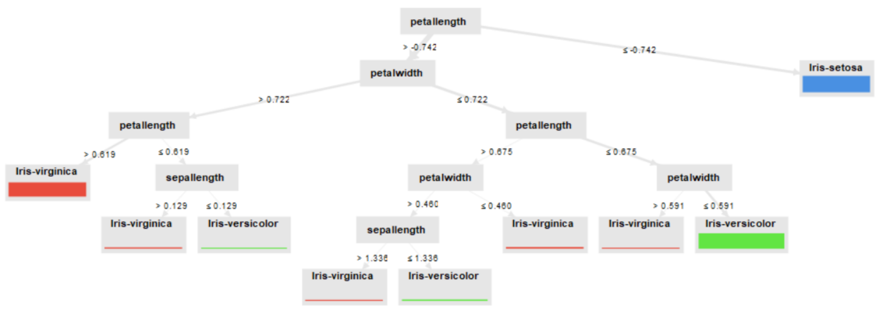


2. Classify the data.

All attributes (petallength, petalwidth, sepallength, sepalwidth)
Pruning: None
Max-Depth: 10

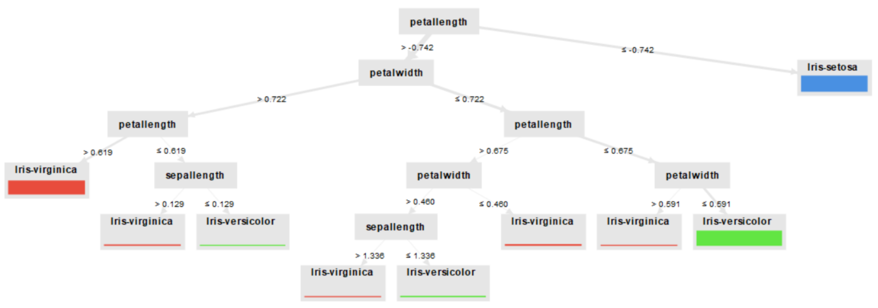
Criterion: **Information gain**

Height: 5
Total number of nodes: 17
Number of leaves: 9
Number of rules:
Iris-setosa: 1
Iris-versicolor: 3
Iris-Virginica: 5



Criterion: **Gini Index**

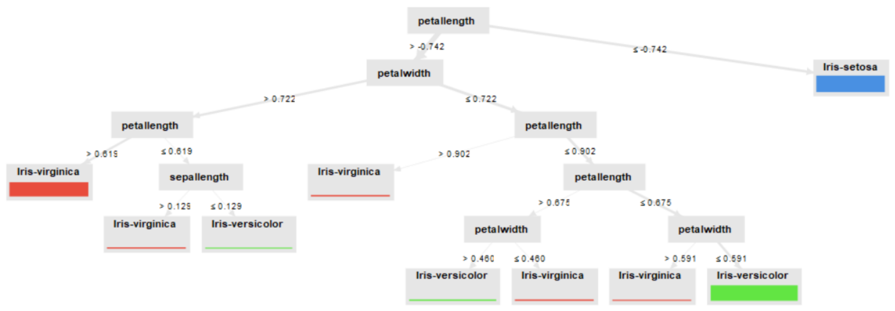
Height: 5
Total number of nodes: 17
Number of leaves: 9
Number of rules:



Iris-setosa: 1
Iris-versicolor: 3
Iris-Virginica: 5

Criterion: **Gain Ratio**

Height: 5
Total number of nodes: 17
Number of leaves: 9
Number of rules:
Iris-setosa: 1
Iris-versicolor: 3
Iris-Virginica: 5



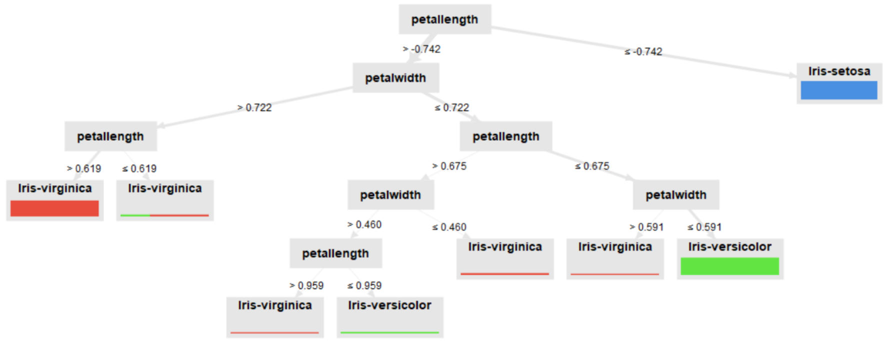
Observations for **All attributes**:

In the tree where all attributes are part of the decision tree, the leafs are all clean and the maximum depth is 5. Between the three trees, only Gain Ratio chooses slightly different splitting nodes. The height, total number of nodes, number of leaves and number of rules are the same in all of them. Only the Gain Ratio uses a fifth decision rule to predict the most instances of Iris-versicolor.
It also appears that petallength and petalwidth outclass attributes that describe sepals. A sepallength attribute appears only at one node in the Gain Ratio tree and twice in both the other trees, and only at the parents of leaf nodes. Sepalwidth is not counted into the splitting process at any point.

Petallength and petalwidth
Pruning: None
Max-Depth: 10

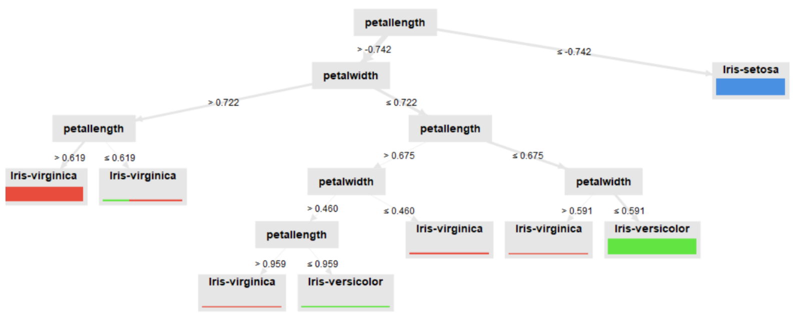
Criterion: **Information gain**

Height: 6
Total number of nodes: 15
Number of leaves: 8
Number of rules:
Iris-setosa: 1
Iris-versicolor: 2
Iris-Virginica: 5



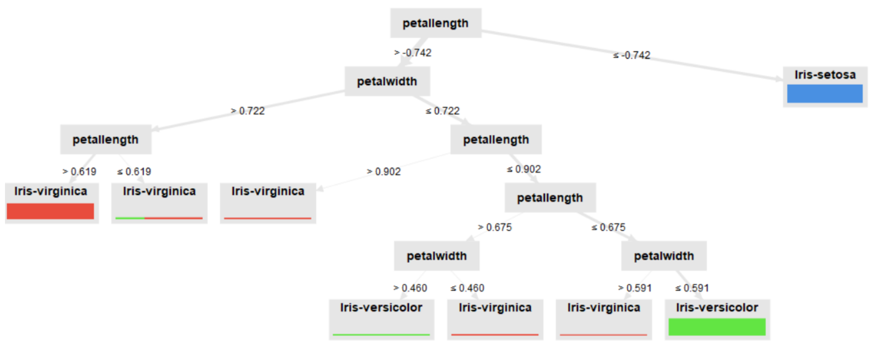
Criterion: **Gini-Index**

Height: 5
Total number of nodes: 15
Number of leaves: 8
Number of rules:
- Iris-setosa: 1
- Iris-versicolor: 2
- Iris-Virginica: 5



Criterion: **Gain Ratio**

Height: 5
Total number of nodes: 15
Number of leaves: 8
Number of rules:
- Iris-setosa: 1
- Iris-versicolor: 2
- Iris-Virginica: 5



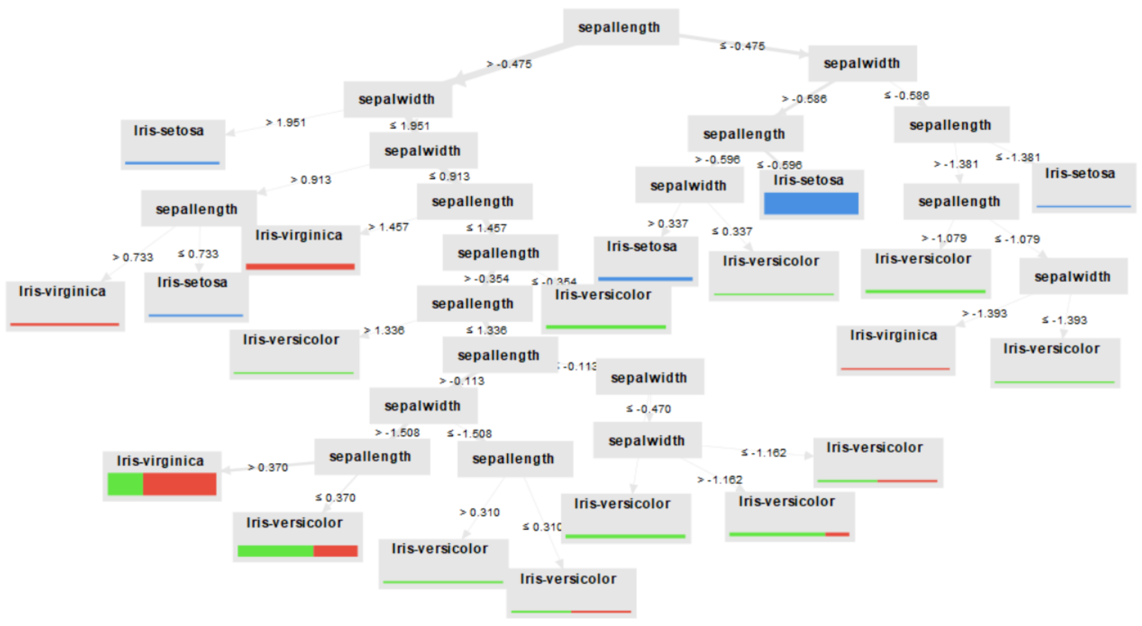
Observations petal:

The tree made out of the attributes petallength and petalwidth has a lot in common with the tree that uses all four attributes. Also here, Gini Index and Information Gain produce the same splitting nodes.
There is one leaf which isn't completely pure. Comparing it to the trees where we used all attributes, it is the exact node where the split happened at the sepallength attribute.
It seems to be that using the petal attributes, the decision tree can predict the classes accurately. There are 50 iris flowers that get fully classified as Iris-setosa after only one branch. There is a branch each for Iris-virginica and Iris-versicolor that achieve a high score of 43 and 47 records being classified as such.

sepallength, sepalwidth
Pruning: None
Max-Depth: 10

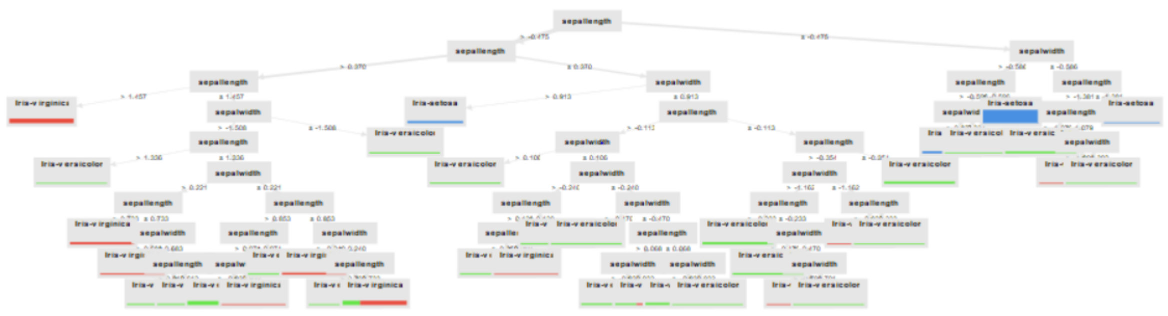
Criterion: **Information gain**

Height: 9
Total number of nodes: 38
Number of leaves: 19
Number of rules:
- Iris-setosa: 4
- Iris-versicolor: 18
- Iris-Virginica: 14



Criterion: **Gini Index**

Height: 9
Total number of nodes: 72
Number of leaves: 33
Number of rules:
- Iris-setosa: 4
- Iris-versicolor: 21
- Iris-Virginica: 12



Conclusion sepal:
Building a decision tree from only the sepal attributes showcases very well what overfitting looks like.
With sepalwidth and sepallength the splitting is performed until the maximum depth and produces a lot more nodes and leaves than the petal attributes. Some of the leaves are not pure. Of all trees, Iris-Setosa is the only one that has pure leaves.
It is apparent now that this model would not be particularly well suited to describe other test data because the rules are too much fitted to the training data.
Being specialized too much on training data, this model would have a low accuracy rate on test data.
Using Gini Index, the tree has the The tree splitted by the gain ratio criterion had the least amount of nodes and thus would have the most model quality, but still isn't as general as the decision tree models made from petal attributes.

3. Look in the book (or somewhere else) what overfitting means and report the meaning. Which trees exhibit overfitting?

Overfitting describes when the model has been specialized too tight to the training data. An overfit model has learned the details of the training data too well and doesn't generalize to new data.
The sepal attributes appear to have a lot of randomness which contributes to the model overfitting. That leads to the tree being split too many times into rules that are specific to the training data. This model will make predictions based on the noise from training data and think of it as a concept to be applied on unseen data.
To detect an overfit model, a model can be tested on training data and test data that was split from an initial dataset. If the model performs a lot better on the training data, it is likely to be overfit.
In order to remove rules that reflect anomalies of training data, tree pruning has to be applied.

4. Read in the book the section “8.2.3 Tree Pruning” p. 344. Explore the possibilities “Apply pruning” and “Apply prepruning” with either “information gain” or “Gini Index”. Give a pruned tree that seems neither overfitted nor underfitted to you and explain how you have generated it with RapidMiner or Jupyter Notebook.

Tree pruning applies statistical measures and hyperparameter tuning to subtract the branches that overfit the decision tree. By removing redundant subtrees the tree loses its complexity and fits better to unseen data.

Prepruning:

Prepruning happens at early construction phase to keep the trees small from the start. Once a halting condition is met, the node turns into a leaf before splitting. It involves tuning the hyperparameters that are used as stopping criteria. To perform prepruning one has to try out a range of values, and the best ones will be fed into the decision tree.

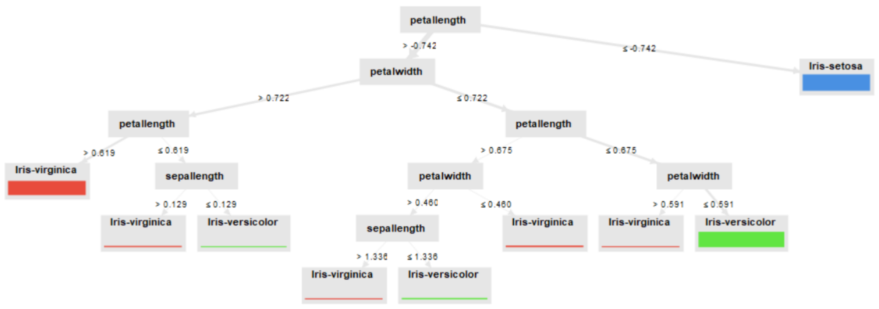
Pruning:

Postpruning (or just pruning) is applied when a tree is fully grown. It removes branches bottom-up and turns inner nodes into leafs when a certain criteria is met. The leaf is labeled with the most frequent class among the subtree. Here the parameters to be tuned are the max-depth and the confidence level, which is used for the pessimistic error pruning approach. In the pessimistic pruning, the tree with the lowest error among possible leaf removals is returned.

DEFAULT SETTINGS:

All attributes with **Gini Index, no pruning, max-depth: 10**

Height: 5
Total number of nodes: 17
Number of leaves: 9
Number of rules:
- Iris-setosa: 1
- Iris-versicolor: 3
- Iris-Virginica: 5

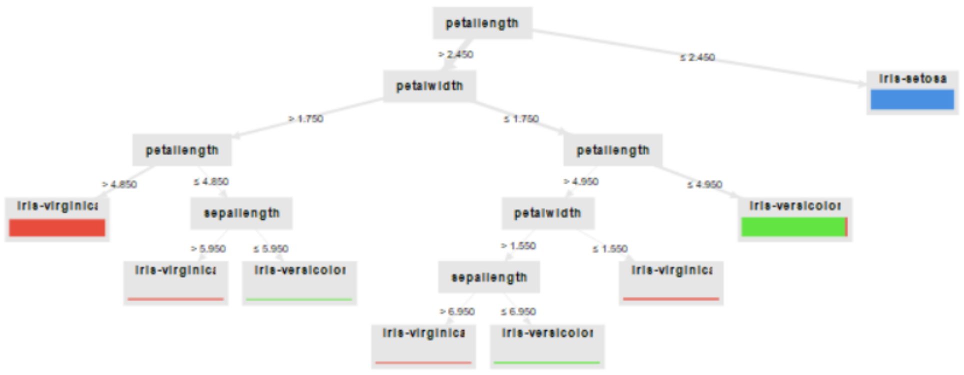


Pruning in RapidMiner

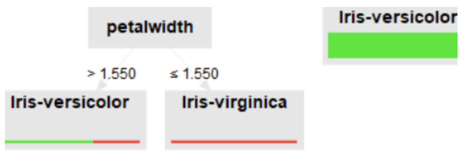
All attributes with **Gini Index, with pruning, max-depth: 10**

Confidence: 0.1

Height: 5
Total number of nodes: 15
Number of leaves: 8
Number of rules:
- Iris-setosa: 1
- Iris-versicolor: 3
- Iris-Virginica: 4



At confidence level < 0.04, there are 9 leaf nodes again which are all pure. Same with confidence level >= 0.2.
If maximal depth < 6, the the leaves decrease to 7, but one leaf becomes more impure.

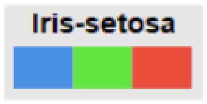


Conclusion:
With only pruning, the best result is achieved by using max depth of >7 and confidence level of 0.1.

Prepruning (no pruning)

In RapidMiner, the prepruning measures to be tuned are:
- Criterion: Gini-Index, Information Gain etc.
- Minimal gain: if the gain for node is greater than the minimal gain, there will be a split. Else the attribute will become a leaf. A higher minimal gain will return a smaller tree,

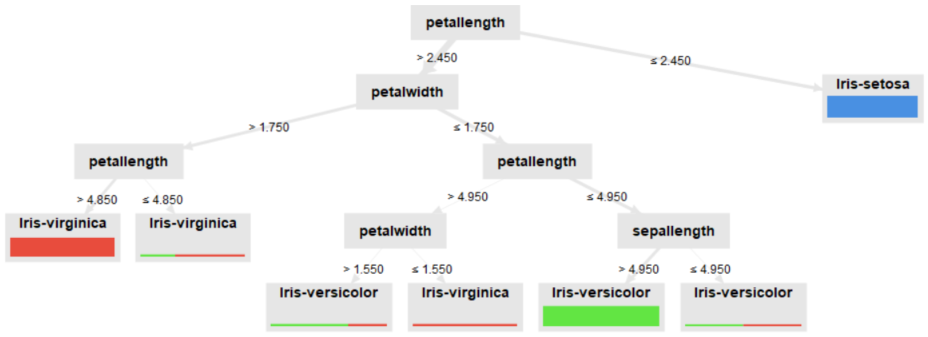
and if the value is too high, a tree with a single node can be generated. As seen here



at value >= 0.6:

- Minimal leaf size: A threshold where the leaf is set to not to have a lower size than the minimal leaf size value.
- minimal size for split: If the number of observations in one child node is below the minimal size for split treshold, the split won't be performed.
- Number of prepruning alternatives: If a node can't split anymore due to prepruning, nodes that are parallel to that node will be tested for splitting.

Default settings
Max-depth: 10
Minimal gain: 0.01
Minimal leaf size: 2
Minimal size for split: 4
Number of prepruning alternatives: 3



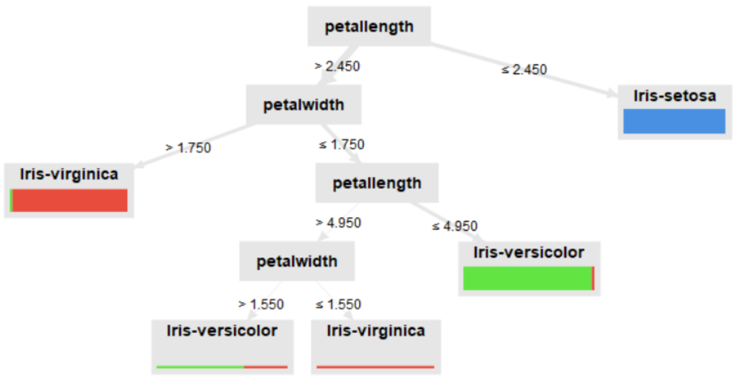
Height: 5
Total number of nodes: 13
Number of leaves: 7
Number of rules:
Iris-setosa: 1
Iris-versicolor: 3
Iris-Virginica: 3

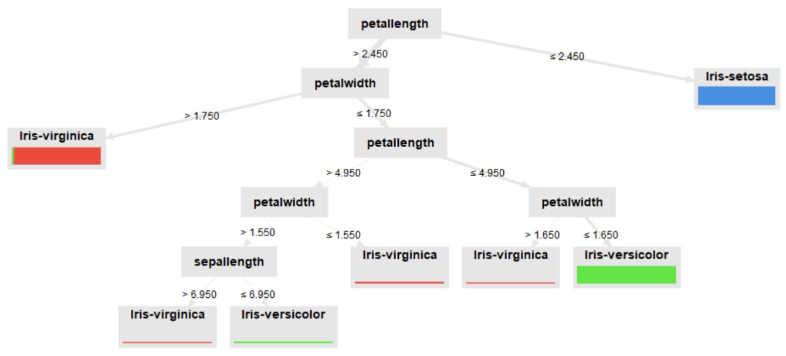
Here the tree has 7 leaves, of which 3 are impure. To better the purity, the minimal leaf size could be tuned to not allow such mixed leaves. To make the tree smaller, the minimal gain could be increased.

The tree decreases to 5 leafs and height 4 by tuning the minimal gain to 0.49.

This is due to 2 inner nodes that didn't split any further at the minimal gain threshold. The class sums up all possible outcomes. This model has the best result so far but still has an impure leaf

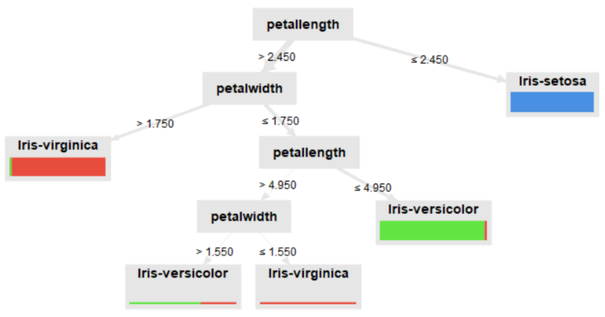
Minimal gain: 0.49:





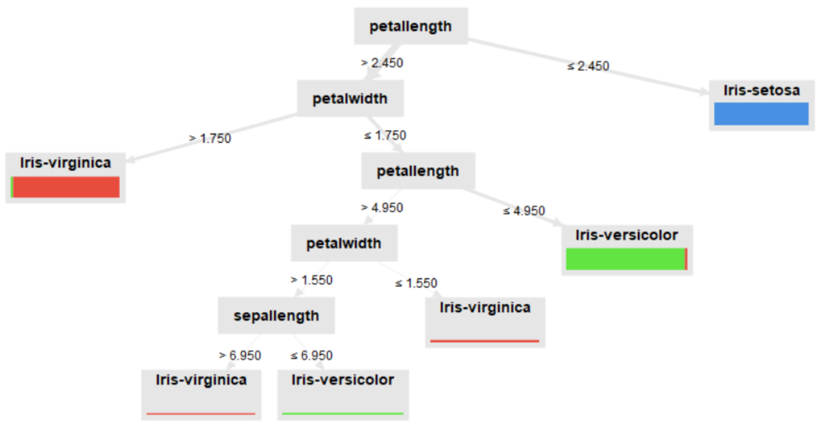
By tuning the minimal leaf size to 1, the leaf nodes are allowed to only have 1 example.

Here the minimal size for split is set to 6. If a node has less than 6 items, another split won't be performed.



A possible decision tree for Iris Dataset

The cleanest tree with the lowest achieved height was produced by following parameters:



All attributes with **Gini Index, pruning & prepruning**
max-depth: 6

Height: 5
Total number of nodes: 10
Number of leaves: 6
Number of rules:

- Iris-setosa: 1
- Iris-versicolor: 2
- Iris-Virginica: 3

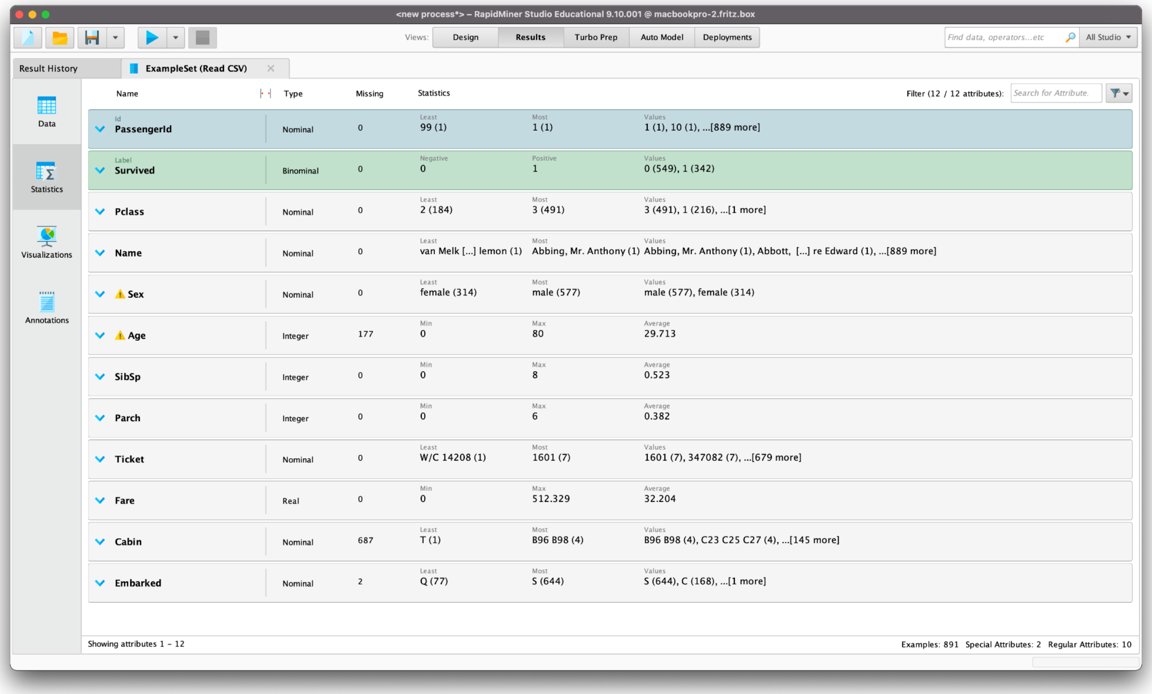
Pruning:
Confidence: 0.1
Prepruning:
Minimal gain: 0.49
Minimal leaf size: 1
Minimal size for split: 3
Number of prepruning alternatives: 3

Conslusion:

The last decision tree does not look certainly balanced but it produces the cleanest and least leaf nodes. There could be better prepruning done on the minimal leaf size and the number of prepruning alternatives could be further assessed to create a tree that is less overfitted at some nodes.

Exercise 3: Decision tree with the titanic dataset

1. Read the dataset. RapidMiner: Pay attention that “Survive” should have the role “label” and the type “binomial”



Name	Type	Missing	Statistics		
Id PassengerId	Nominal	0	Least 99 (1)	Most 1 (1)	Values 1 (1), 10 (1), ...[889 more]
Label Survived	Binominal	0	Negative 0	Positive 1	Values 0 (549), 1 (342)
Pclass	Nominal	0	Least 2 (184)	Most 3 (491)	Values 3 (491), 1 (216), ...[1 more]
Name	Nominal	0	Least van Melk [...] lemon (1)	Most Abbing, Mr. Anthony (1)	Values Abbing, Mr. Anthony (1), Abbott, [...] re Edward (1), ...[889 more]
⚠ Sex	Nominal	0	Least female (314)	Most male (577)	Values male (577), female (314)
⚠ Age	Integer	177	Min 0	Max 80	Average 29.713
SibSp	Integer	0	Min 0	Max 8	Average 0.523
Parch	Integer	0	Min 0	Max 6	Average 0.382
Ticket	Nominal	0	Least W/C 14208 (1)	Most 1601 (7)	Values 1601 (7), 347082 (7), ...[679 more]
Fare	Real	0	Min 0	Max 512.329	Average 32.204
Cabin	Nominal	687	Least T (1)	Most B96 B98 (4)	Values B96 B98 (4), C23 C25 C27 (4), ...[145 more]
Embarked	Nominal	2	Least Q (77)	Most S (644)	Values S (644), C (168), ...[1 more]

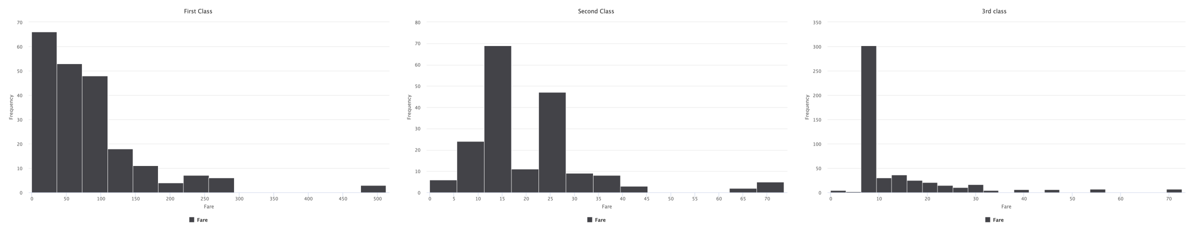
Showing attributes 1 – 12
Examples: 891 Special Attributes: 2 Regular Attributes: 10

2. Explore the data with the summary statistics and different visualizations. Decide and justify which attributes you will keep to predict whether a person will survive. Pay a special attention to attributes with missing values and explain how you are going to handle them.

The summary statistics (see above) already shows missing values for the features *Age*, *Cabin* and *Embarked*.
Further manual investigation into the cleanliness of the dataset supported this, but also showed that values for *Fare* contain 15 instances of 0 (and also some other unusual low values (outliers) like #873, a first class passenger who apparently only paid $5 for his ticket), indicating erroneous or missing data here as well.
Cabin not only has a lot of missing values, many are also not atomical, meaning that there are multiple values for one passenger. The usefulness of the feature *Cabin* is further detailed below.
The attribute *Ticket*, despite having a lot of differently formatted values (see below as well), has 4 values “LINE”, which seem to be erroneous or missing as well.
All other attributes seem to contain clean values. Furthermore there is no duplicate or obsolete data, thus is the dataset otherwise clean.

Except for manually researching and filling them manually, which may often be possible although time consuming, we suggest the following ways to treat the **attributes with missing or erroneous values**. (The strategy that we used in the end is highlighted)

- ***Age*****: Global median of 28** (Since *Age* has a quite symmetric distribution, the average of 29.713 may be used as well). We do not suggest removing all 177 samples, since they make up almost a fifth of the whole dataset. It may be interesting to compare the decision trees for both cases, however.
- ***Ticket*****:** Constant the algorithm ignores or **disregarding the feature altogether.**
- ***Fare*****: Median for samples belonging to the same class in *Pclass*,** since we already know from context that this attribute best classifies ticket prices. Since the data is extremely skewed (see below), the mean should not be used. The feature may also be ignored altogether.
- ***Cabin*****:** Constant the algorithm ignores or **disregarding the feature altogether.**
- ***Embarked:*** Constant the algorithm ignores, disregarding the feature altogether or **deletion of the two affected tuples,** as they only make up 0.2% of the dataset.
-



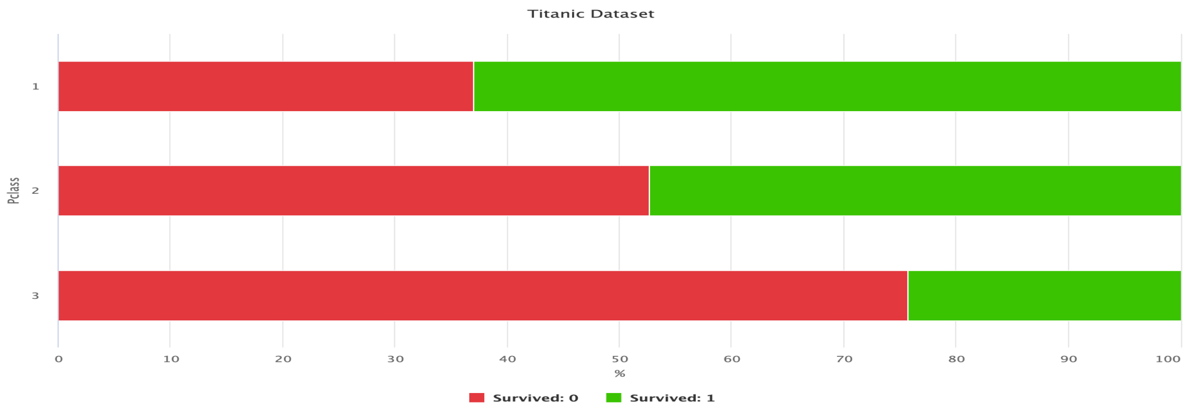
Fare: 1st class avg≈84, med≈60 | 2nd class avg≈21, med≈14 | 3rd class avg≈14, med≈8

The attributes *Sex* and *Age* are flagged by RapidMiner as potentially leading to biased models. In this specific case the warning can be neglected, since we are not trying to predict survivors of future maritime disasters.
Unfortunately this dataset does not contain an attribute whether a passenger was able to get onto a lifeboat like other versions of the titanic dataset. Based on descriptions of the disaster, however, we can assume that this feature would be highly correlated with the feature *Survived*, as no one survived who wasn't on a lifeboat.
PassengerId, *Name* and *Ticket* are almost definitely not features that can be used for classification, as they are unique or almost unique to every passenger. As many are not numerical, *Ticket* values are also not ordinal (without further information to decode them), hence (arguably futile) patterns like "Holders of ticket numbers x to y had a higher survival rate than the mean" can't be found. But they may be useful to identify groups or families. Same goes for the attribute *Cabin*. It may be possible to cluster the values and get an understanding on which part of the ship there was a higher or lower chance of survival (provided that a systematic/ordinal naming scheme was used, which is very likely). But since there are over 77% of values missing for this attribute, we highly doubt the validity of such an analysis, as the available data may not be a good sample.

In conclusion we suggest based on the summary statistics and historical information the most interesting attributes for survival prediction to be *Pclass* (Passenger Class), *Sex*, *Age*, *SibSp* (Number of siblings or spouses on board), *Parch* (Number of Parents or Children on board).
PassengerId, *Name*, *Ticket* and *Cabin* (in its current form, as detailed above) are definitely not useful and the benefit of *Fare*, and *Embarked* needs to be seen.

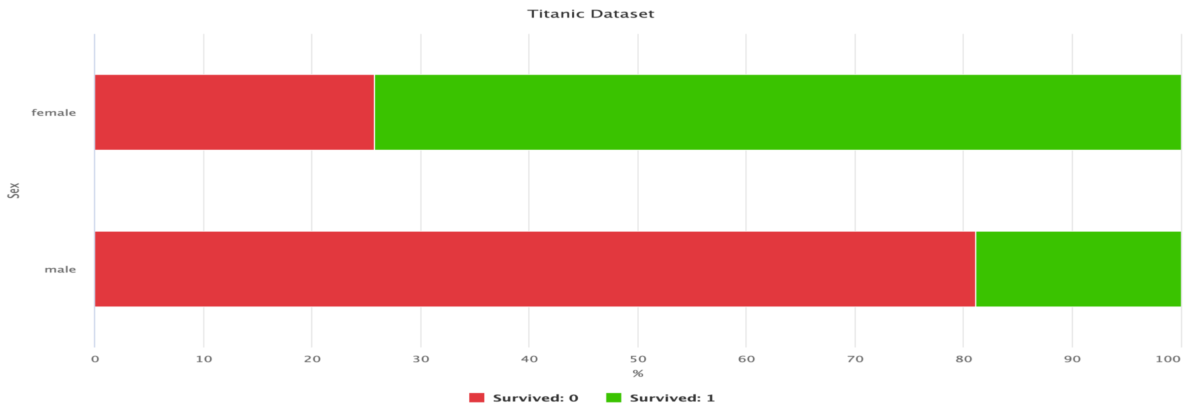
We use visualizations to determine which attributes out of *Pclass*, *Sex*, *Age*, *SibSp*, *Parch, Fare* and *Embarked* we are going to keep to predict whether a persion will survive.

Pclass



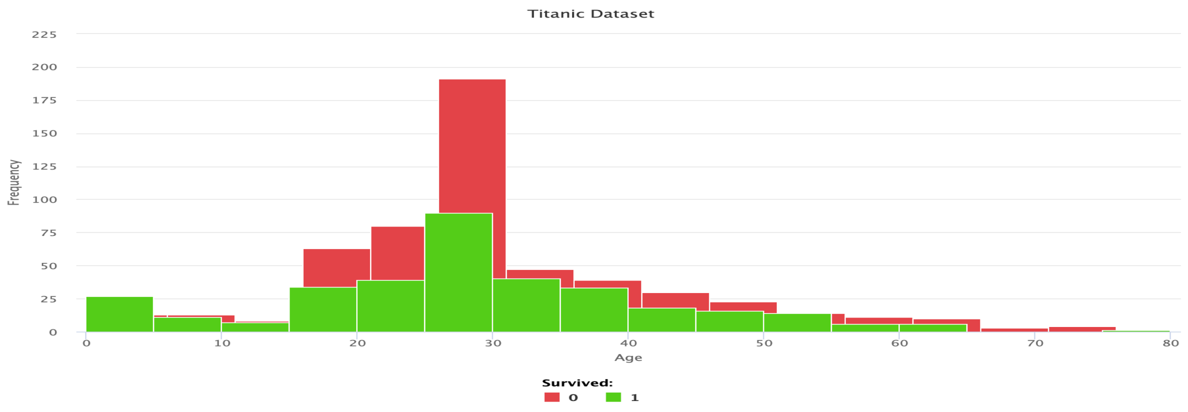
There are noticeable differences in survival rate across the three passenger classes. This feature seems to be a good predictor.

Sex



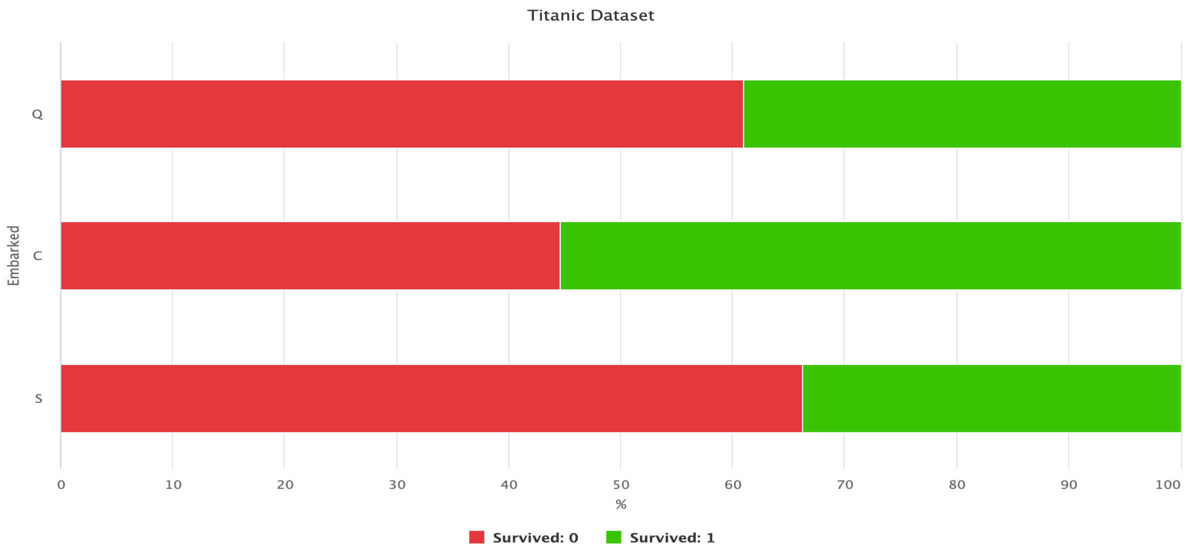
There is a major difference in survival rate depending on gender. This feature seems to be a good predictor.

Age



The amount of passengers who survived and the amount of passengers who did not follow a similar shape across the different age groups. At first glance this feature does not seem to be a good predictor, but we will include it in our model nonetheless since there are indicators that suggest otherwise: More very young children and passengers around 50 years old survived than drowned and unexpectedly many young adults under 25 did die.

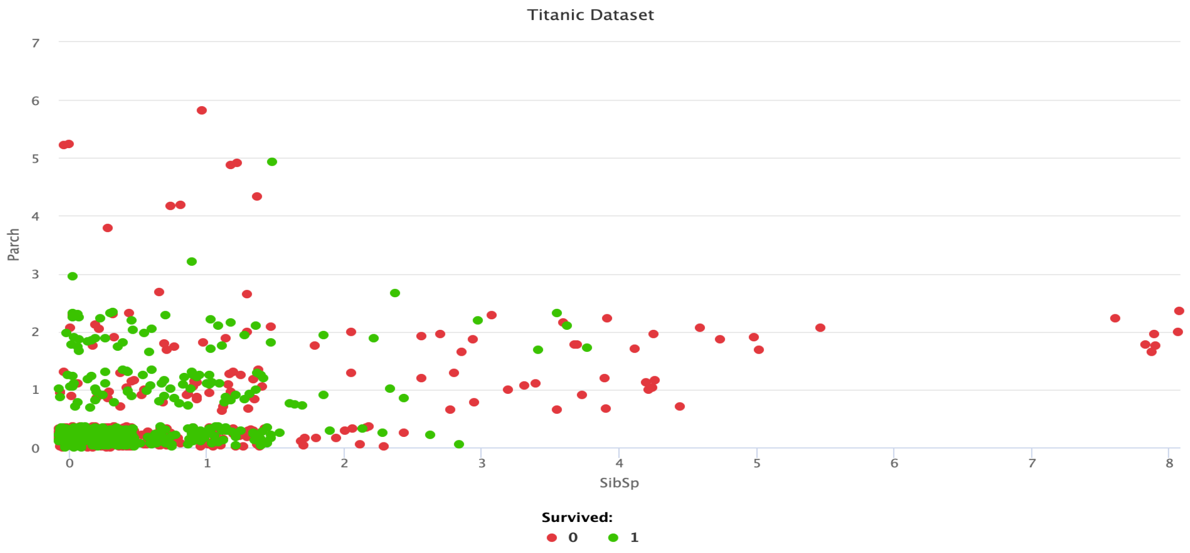
Embarked



Since this feature is not ordinal and the share of survivals vs. deaths is quite close across all three values, this feature might not be a good predictor. An embarkment in Cherbourg however seems to suggest a slightly better chance of survival. Whether a French origin predicts a more likely survival compared to a British one (Embarkment in Southampton or Queenstown) seems highly coincidental. This assumption should still be tested, however, as it could not only indicate an interesting correlation with other features like *Pclass* or *Cabin*, but also lead to a better decision tree model.

The following scatter plots have an increased "Jitter" value to make the individual samples more visible. This explains why data points for integer values don't exactly sit on the line.

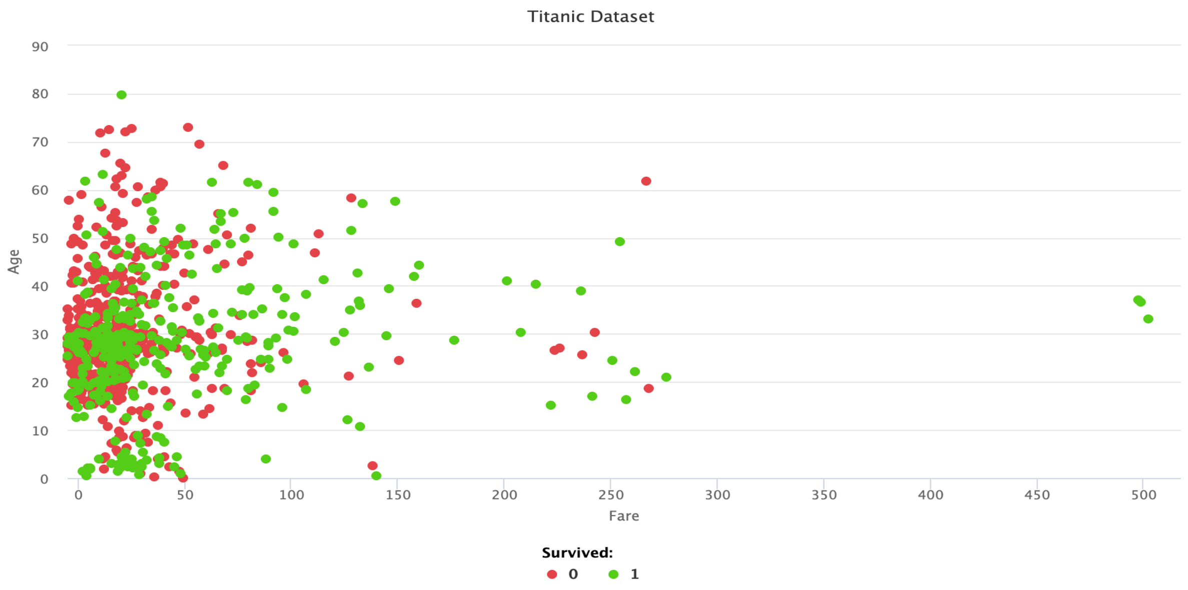
SibSp and Parch



Family could be a useful predictor, as passengers with more than four family members of the same generation (sibling and/or spouse) did not survive. Similarly almost all people with more than four members of a different generation (parent and/or children) did not survive.

Having between one and three children or parents on board, however, seems to have increased one's chance of survival.

Fare and Age



As expected from the visualization of the *Pclass* feature above we see more survivals than deaths when passengers paid higher fares (more than around $100) and conversely more deaths of passengers who paid less than roughly $50. The *Age* attribute was mainly included to facilitate a broader scattering of the data points, but similar conclusions to the respective histogram above can be drawn from this representation as well.

We will include all features listed above in our model, namely *Pclass, Sex, Age, Embarked, SibSp, Parch* and *Fare,* but compare it to a model only based on *Pclass, Sex* and *Age,* as the visualizations suggest that these are the most useful predictors for survival.

3. If a predictor always predicts "will not survive", what will be its probability to be correct? Explain your answer.

The predictor always predicts "will not survive", hence P'=0 and N'=891 (all passengers). Out of those, 549 truly did not survive (TN) and 342 actually did survive and are falsely predicted to have died (FN). Since P'=0, TP and FP also equal 0. The probability to be correct equals the overall chance to have not survived (around 61,62%), as the predictor will only be correct in those cases and in no other case. The confusion matrix is as follows:

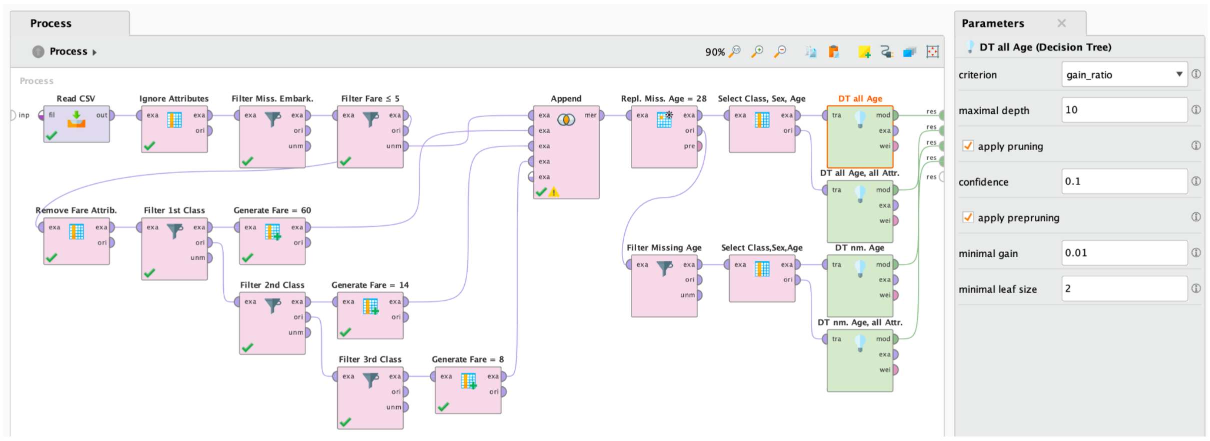
	Predicted survived	Pred. not survived	
Actual survived	**0** (TP)	**342** (FN)	**342** (P)
Actual not survived	**0** (FP)	**549** (TN)	**549** (N)
	0 (P')	**891** (N')	**891** (P+N = P'+N')

The predictor's probability to be correct or its accuracy is calculated like this:

$\frac{TP+TN}{P+N} = \frac{0+549}{342+549} = \frac{549}{891} \approx 0.6162 = 61,62\%$

4. Run the decision tree algorithm. List the rules for the 5 leaves with the biggest number of elements. If your tree has less than five leaves, list the rules for the leaves that are in the tree. What do these rules tell us?

RapidMiner process (including data cleaning as described above) for a total of four decision trees (missing ages replaced, examples with missing ages removed, all attributes and only *Pclass, Sex* and *Age*) and **default parameters for the decision tree algorithm**

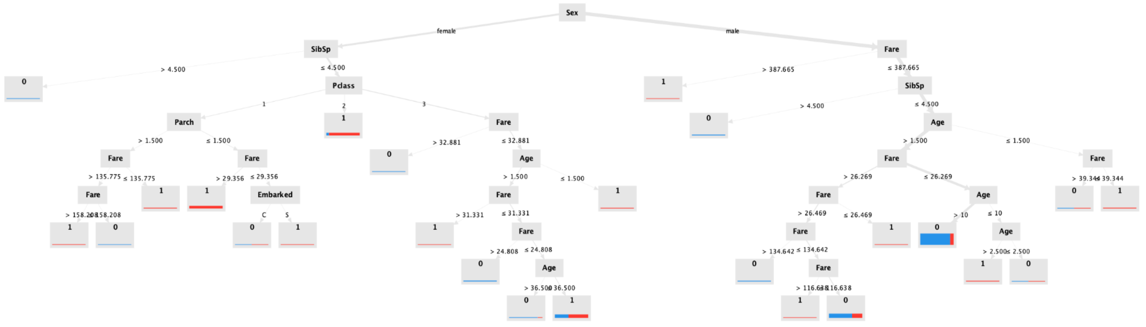


Parameters
DT all Age (Decision Tree)

criterion	gain_ratio
maximal depth	10
✓ apply pruning	
confidence	0.1
✓ apply prepruning	
minimal gain	0.01
minimal leaf size	2

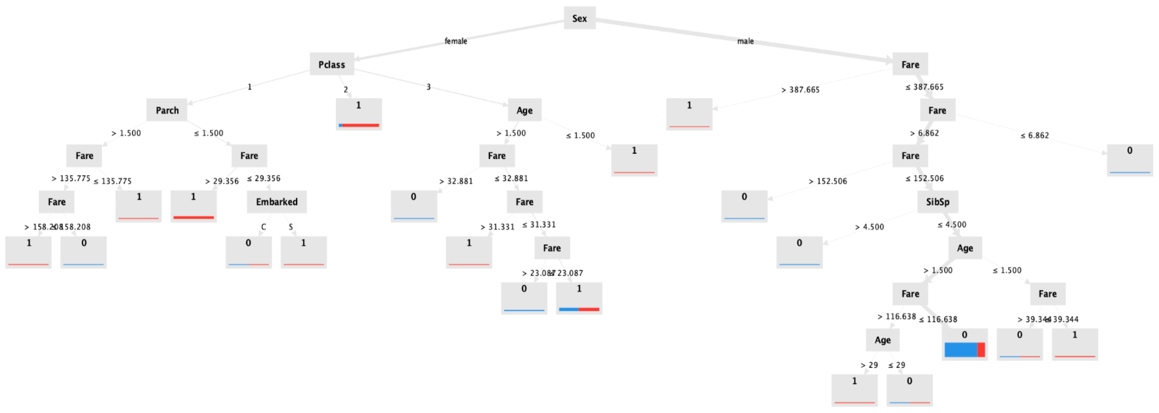
Decision tree 1

Missing *Age* values replaced, all attributes (except *PassengerId*, *Name*, *Ticket* and *Cabin*)

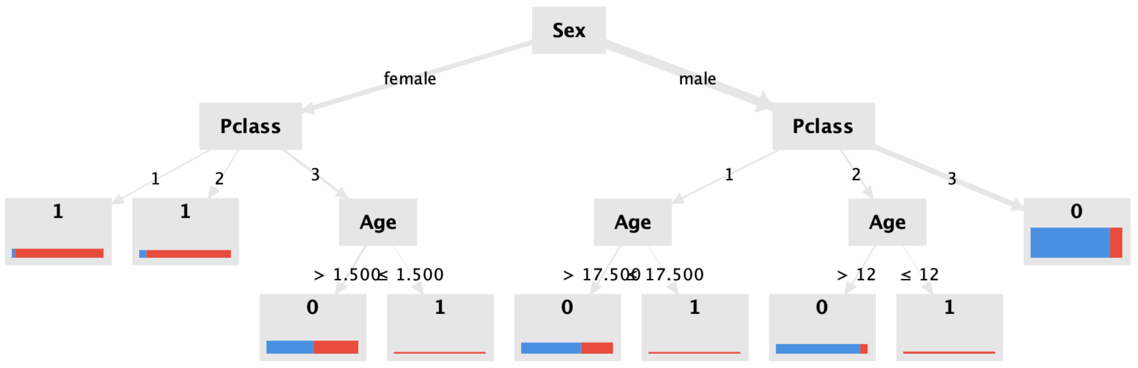


Decision tree 2

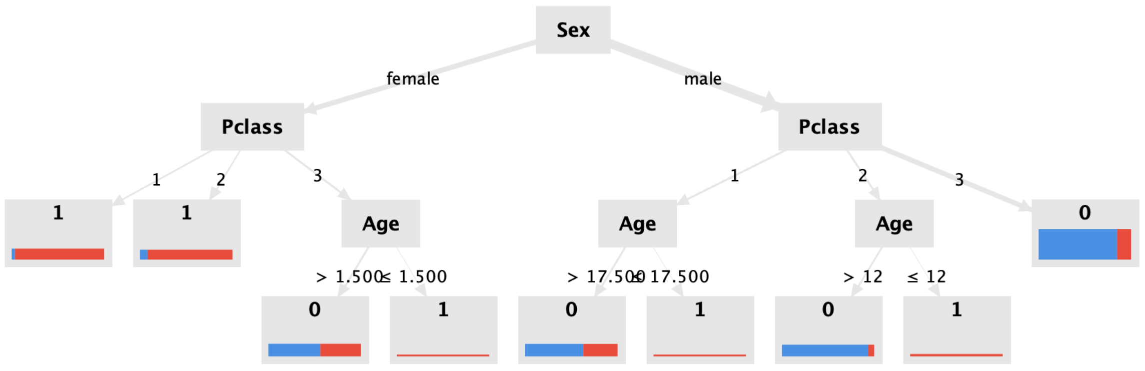
Examples with missing *Age* values removed, all attributes (except *PassengerId*, *Name*, *Ticket* and *Cabin*)



Decision tree 3
Missing *Age* values replaced, only attributes *Pclass, Sex* and *Age*



Decision tree 4
Examples with missing *Age* values removed, only attributes *Pclass, Sex* and *Age*



Right from the start we can see that the trees based on all attributes got extremely complicated and we suspect an unfavorable overfitting of the data by the model. We will continue only with the two trees based on the three attributes *Pclass, Sex* and *Age* and tried tweaking the hyperparameters as some leaves are still very impure, but our trees got very deep very fast (meaning we couldn't find a "sweet" spot that would not overfit and still lead to purer leaves). Hence we kept the default parameters (see subtask 5 as well). This suggests that for many the chance of survival did not depend on the analyzed features. Conclusions on the chance of survival can be drawn from this model nonetheless, as detailed below. Comparing the decision tree 3 and 4, the one where we used all examples and replaced missing *Age* values with 28 (decision tree 1) led to marginally purer leaves than the one where we removed the examples altogether. But we think either strategy would have been fine in the end. Their whole description is on the following page.

Decision tree 3

```
Sex = female
|   Pclass = 1: 1 {0=3, 1=89} (92 elem., 89/92 survival prob.)
|   Pclass = 2: 1 {0=6, 1=70} (76 elem., 70/76 survival prob.)
|   Pclass = 3
|   |   Age > 1.500: 0 {0=72, 1=68} (140 elem., 68/140 s. prob.)
|   |   Age ≤ 1.500: 1 {0=0, 1=4} (4 elem., 4/4 survival prob.)
Sex = male
|   Pclass = 1
|   |   Age > 17.500: 0 {0=77, 1=41} (118 elem., 41/118 s. prob.)
|   |   Age ≤ 17.500: 1 {0=0, 1=4} (4 elem., 4/4 survival prob.)
|   Pclass = 2
|   |   Age > 12: 0 {0=91, 1=8} (99 elem., 8/99 survival prob.)
|   |   Age ≤ 12: 1 {0=0, 1=9} (9 elem., 9/9 survival prob.)
|   Pclass = 3: 0 {0=300, 1=47} (347 elem., 47/347 survival prob.)
```

Decision tree 4

```
Sex = female
|   Pclass = 1: 1 {0=3, 1=80} (83 elem., 80/83 survival prob.)
|   Pclass = 2: 1 {0=6, 1=68} (74 elem., 68/74 survival prob.)
|   Pclass = 3
|   |   Age > 1.500: 0 {0=55, 1=43} (98 elem., 43/98 s. prob.)
|   |   Age ≤ 1.500: 1 {0=0, 1=4} (4 elem., 4/4 survival prob.)
Sex = male
|   Pclass = 1
|   |   Age > 17.500: 0 {0=61, 1=36} (97 elem., 36/97 s. prob.)
|   |   Age ≤ 17.500: 1 {0=0, 1=4} (4 elem., 4/4 survival prob.)
|   Pclass = 2
|   |   Age > 12: 0 {0=84, 1=6} (90 elem., 6/90 survival prob.)
|   |   Age ≤ 12: 1 {0=0, 1=9} (9 elem., 9/9 survival prob.)
|   Pclass = 3: 0 {0=215, 1=38} (253 elem., 38/253 survival prob.)
```

(For simplicity's sake we will continue to only analyze decision tree 3)
The rules for the five biggest leaves are:

- Male, 3rd class (347 elements)
- Female, 3rd class, over 1.5 years old (140 elements)
- Male, 1st class, over 17.5 years old (118 elements)
- Male, 2nd class, over 12 years old (99 elements)
- Female, 1st class (92 elements)

These rules tell us that the most important deciding factor was a passenger's sex, followed by their class. With 1st and 2nd class passengers we then see a clear distinction between adults and (young) children. In 3rd class age does not seem to have been as important.

5. For each of these nodes, give the probability of the prediction. Calculate it as follows: Consider as an example a node with 11 elements Survived=1 and 4 elements Survived=0, then the probability Survived=1 of each element of the node is 11/15. Report the parameters that you have used to build the tree (criterion for the split, pruning and so on).

The probability to have survived for the five biggest leaves are:

- Male, 3rd class: 47/347 ≈ 13,54%
- Female, 3rd class, over 1.5 years old: 68/140 ≈ 48,57%
- Male, 1st class, over 17.5 years old: 41/118 ≈ 34,75%
- Male, 2nd class, over 12 years old: 8/99 ≈ 8,08%
- Female, 1st class: 89/92 ≈ 96,74%

Out of these five categories, males above the age of 12 (adult or supposedly adult looking) were most likely to die. Male 1st class passengers had significantly better chances than those in 2nd and 3rd class. It seems as if male 3rd class passengers were more likely to have survived than those in 2nd class, but 3rd class male passengers are not split by age in this model, hence this leave likely includes children as well. Furthermore we must keep in mind that most of the missing *Age* values were from 3rd class passengers, as well, and are represented as age 28 in the cleaned dataset.
Female passengers on the other hand had way better chances of survival than males of the same class. It is striking that almost all female 1st class passengers survived, but only half of those travelling in 2nd class did.
Including the leaves with fewer elements as well we can support the infamous rule “women and children first” that was used to fill the rescue boats, according to the witnesses of the tragedy. But a passenger’s class was an almost equally important factor to ones survival, as well.

The parameters we used to build the trees were:

- Criterion: gain_ratio
- Maximum depth: 10
- Confidence: 0.1
- Minimal gain: 0.01
- Minimal leaf size: 2