Luleå University of Technology (LTU)

Department of Computer Science, Electrical and Space Engineering

Digital Services and Systems

COURSE: D0025E

Data Mining Assignment: DECISION TREES

MEMBERS: Oscar Arandes Tejerina, Anders Dehlbom,
Erik Vodopivec Forsman, Mattias Konnebäck, Nicole Aguiar Jakobsson,

LP1-2025

Abstract

The report uses the CRISP-DM framework to create a model that determines whether someone would have died on titanic or not. This is done by using a premade dataset that then had some preprocessing done to it (mainly dealing with missing values and removing columns). There was also a new column added that showed what happened to each passenger. When that was done, a Decission Tree model was then created in both Knime and AiStudios (RapidMiner). The models had high confidence scores. These models were trained on a training dataset and then tested on different data (including data representing people known to us).

Contents

Chapter 1

The CRISP-DM Process

In this chapter, we provide a detailed overview of the CRISP-DM process and its key phases [She00].

1.1 Phase I: Business Understanding

The focus of this project is to create a model that can determine whether a person would have survived on the titanic or not. To achieve this, a dataset containing information about all passengers is required. The plan for doing this entails using decission trees to determine if a person would survive on the titanic or not. Data about the actual passengers are supposed to be used to train and test the model. When that is done the model can be used to make determinations about if other people would have survived or not. A model will be created in both AiStudios (RapidMiner) and Knime.

1.2 Phase II: Data Understanding

The training data can be downloaded as a zip file in CSV format from the following link while the test data can be downloaded from this other link.

The dataset has information about the passengers on titanic. The data covers features about the passengers only, not crew. The attributes in the data are the following. The features `name`, `sex` and `age` refer to the name, sex and age of the passenger. Further personal information includes `sibsp`, the number of siblings and/or spouses aboard, `parch`, the number of parents and/or children aboard. Specifics of the way the passengers travel are reflected in `class`, the passenger class (1 = 1st; 2 = 2nd; 3 = 3rd), `ticket`, the ticket number, `cabin`, the cabin number and the port of embarkation in `embarked` (C = Cherbourg; Q = Queenstown; S = Southampton) and `home.dest`, describing the home of the passengers and their destination. The survival of the tragedy is encoded in the attributes `survival` (0 = No; 1 = Yes), `boat`, the lifeboat number (if survived) and `body`, the body number (if they did not survive and the body was recovered). The amount of rows in the dataset is approximately 1,300.

There is a lot of missing data in this dataset. Examples of columns with a lot of data missing is `cabin`, `boat` (lifeboat), and `body`. The lifeboat values are missing due to someone not having had a spot on one of the lifeboats, and the `body` ones are missing due to the body never being found and the person being presumed dead (yet not confirmed dead). The `cabin` values simply seems to be unavailable.

1.3 Phase III: Data Preprocessing

In order to analyze the data with our decision tree algorithm, we need to prepare the data in the following way. The first step is to remove the last row of our dataset, which appears to be a row with all missing values. After this minimal change, we aim to create a new attribute `Survival_Result` that captures whether the person *Died* (and their body was found), *Lost* (the body was never found but assumed dead), or *Lived.* This means we need to combine the information in the `survived` attribute (survived or not) with the `body` attribute (which assigns a body number for those bodies recovered).

Next, we want to filter out the `survived` and `body` attributes (since the information is now encoded in the newly created feature `Survival_Result`). We also want to remove all features that will not be predictive in determining whether a person survived or not in our test dataset. These include the attributes `name`, `ticket`, `cabin` (it could be an interesting predictive attribute, but it has almost all values missing), and `home.dest`.

It is now time to deal with the missing values in our dataset. We observe two missing values in `embarked` and one missing value in `fare`, which we do not mind since decision trees can handle a few missing values. The part that needs our focus is the large percentage of missing values, about 20% of the total dataset, in the `age` attribute. We decide to substitute the missing values with the *median* of the distribution [Sil24], since we observe that the age distribution is skewed (see Fig. 1.1). See also Chapter 2 for different approaches to achieve data imputation in RapidMiner and KNIME. Now our data is ready to be input into the decision tree algorithm.

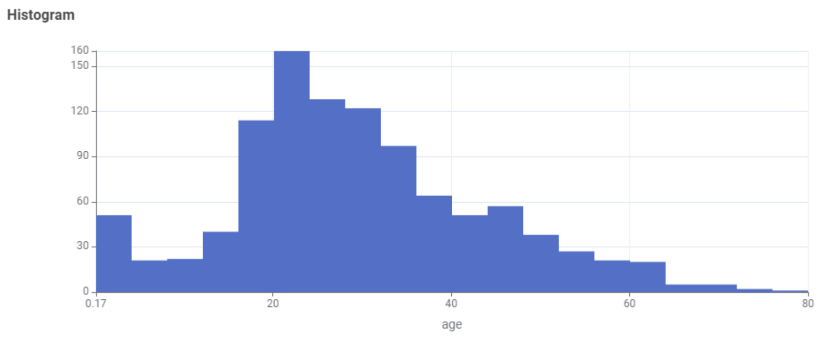


Figure 1.1: Skewed distribution for the `age` feature.

1.4 Phase IV: Modeling

1.4.1 Modeling Phase

The algorithm used in this analysis is called a decision tree algorithm. Considering that the data set contained both categorical and numerical data this was deemed the best approach. The decision tree algorithm is a non-parametric but supervised learning algorithm. Supervised learning means that the data the algorithm trains on is labeled. In this case the label used was `Survived_Result` with the parameters: `died`, `lost`, and `survived`. The model is supervised because it learns from already known outcomes.

The non-parametric part refers to that the model doesn't adhere to specific structures before learning. For example $a + b = c$. The non-parametric model instead adapts to the data by observing patterns and splitting the data into smaller and smaller groups, in this case presented as a tree like structure.

Different parameter settings have varying impact on how the Decision Tree is structured. One big difference appears to be when criterion is changed from Information Gain to Gain Ratio. Gain Ratio is calculated as Information Gain/Split Information that penalises attributes that create many small branches. Shifting from Information Gain to Gini does not produce any significant difference. Modifying tree depth and minimum leaf size also affects the trees complexity. Reducing the maximum depth (for example from 10 to 5) makes the model stop splitting earlier, producing a simpler tree with broader categories at the leaves.

Based on these different tests, the final configuration used was:

- `criterion = Gain Ratio`
- `max depth = 8`
- `min leaf size = 4`
- `pre-/post-pruning = enabled`

The attribute `fare` was also excluded this is a diversification of `pclass`. This produced a compact and interpretable tree to visualise an overview of the main survival factors of Titanic.

1.5 Phase V: Evaluation

The purpose of this assignment was to evaluate whether the decision tree model could predict survival outcomes and identify the most important factors leading to these outcomes.

The confidence values were consistently high for the predictions, and all confidence values close to (0.667) and above 0.7 indicating a strong correlation. This shows that the model can distinguish properly between `survived`, `lost` and `died` passengers.

The model was also applied to a smaller dataset containing individuals that were known to us beforehand and predictions showed that the model could accurately determine their survival status. This shows that the model can take new data and predict their survival outcomes.

All in all, the model meets the business criteria and goals of the project and no major issues were identified.

1.6 Phase VI: Deployment

The deployment phase has been effectively completed through this report.

Chapter 2

Tools Insights

This chapter presents a comparative analysis of two data mining tools in terms of ease of use, efficiency, and cost to solve Assignment 4 in the course.

2.1 RapidMiner Insights

This section explains our insights while using RapidMiner (Altair AI Studio).

2.1.1 Ease of Use

Let us go through the different operators used to run the decision tree model on our dataset (see Fig. 2.1).

First, we apply two *Retrieve* operators, one for the training dataset and one for the test dataset. In the training branch, the *Filter Examples* operator is used to remove single empty rows containing only missing values.

We also have to separately handle missing values for the attribute `age`, and we, therefore, use the *Aggregate* operator to compute the median value for column `age`. We then rename the result, the attribute `median(age)`, using operator *Rename*, giving it the name `age_median`. Next, we use the *Set Macro* operator to store this median value as a process variable (with macro name `median_age` and value name `age_median`). And, finally, we use the *Cartesian Product* operator, in order to combine the original dataset with the single-row table containing the calculated median value.

Then, in the *Generate Attributes* operator we perform two key transformations. First, we create a new target column named *Survival_Result* based on the attributes `survived` and `body`. For that, we use this code expression:

```
if(survived == 1, ‘‘survived’’,
    if(! missing (body), ‘‘died’’, ‘‘lost’’)
)
```

Secondly, we create another new column, `age_new`, where missing values in `age` are replaced by the process macro `median_age`. Non-missing values remain unchanged. Here, we use this expression:

```
if(missing(age), eval(\%{median\_age\}), age)
```

In next step, we use operator *Select Attributes* to remove non-predictive and redundant variables such as `name`, `ticket`, `boat`, `body`, `survived`, and other fields. We also have to apply a second *Rename* operator which renames `age_new` back to `age`, in order to have a consistent model input.

Now, we are ready to define the attribute `Survival_Result` as the label attribute. For this, we use the operator *Set Role*.

Finally, the *Decision Tree* operator is used to train the model. In this operator, we configure (and alter) parameters/criteria such as `gain ratio` or `gini index`, we set the `minimal leaf` `size` and the `maximal depth` (of the tree), and we also decide to use both pruning alternatives available.

The *Apply Model* operator takes the trained model (via input `mod`) and our external test dataset (via input `unl`) and generates a predicted survival result for each instance of the test dataset.

This now labeled test set with predictions is connected to *Result* for viewing. For example, it displays the survival predictions for our group members (see Fig. 2.2).

In parallel, we also use the operator *Weights to Data* (connected to the output from *Decision Tree*), in order to get a first, quick overview of the importance of the attributes in the training dataset/model.

2.1.2 Efficiency

Once again, RapidMiner proves to be an effective tool. The complex process of computing different algorithms when deciding on a decision tree model is very user friendly.

On the other hand, the preprocessing this time turned out to be much more complicated. For example, there seems to be no efficient way to easily single out the median value of an attribute, so we had to apply a "work around".

Still, once the dataset is finalized and the model is applied and running, RapidMiner offers a multitude of ways to analyze the result.

2.1.3 Costs

RapidMiner requires a license key and the process for installing it is complicated. Nevertheless, there are free-of-charge licenses for students.

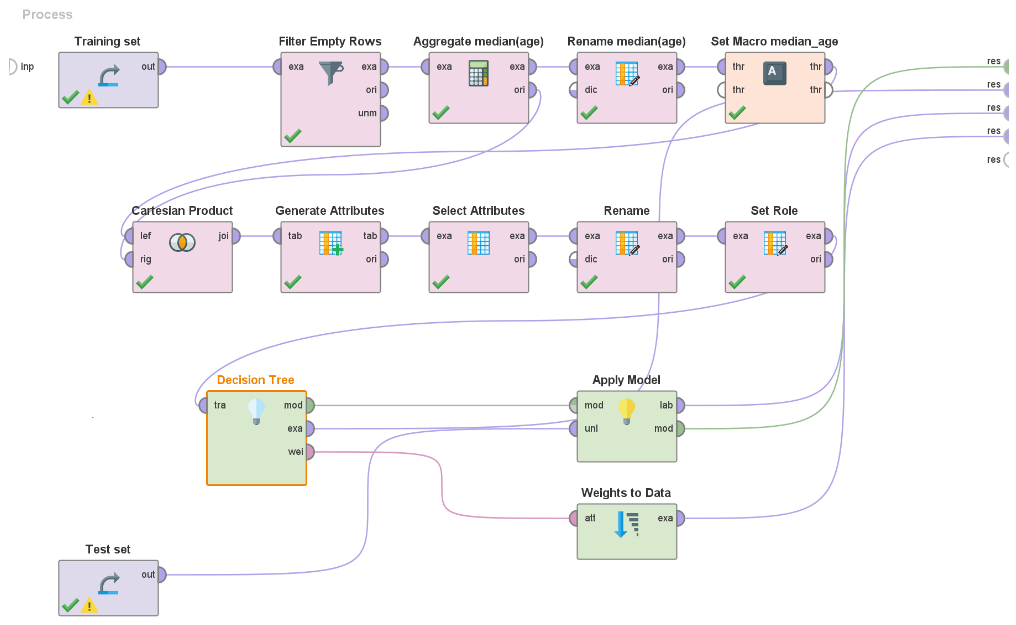
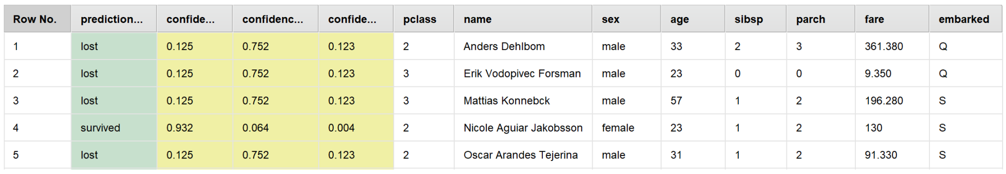


Figure 2.1: RapidMiner process for the decision tree algorithm.



Row No.	prediction...	confide...	confidenc...	confide...	pclass	name	sex	age	sibsp	parch	fare	embarked
1	lost	0.125	0.752	0.123	2	Anders Dehlbom	male	33	2	3	361.380	Q
2	lost	0.125	0.752	0.123	3	Erik Vodopivec Forsman	male	23	0	0	9.350	Q
3	lost	0.125	0.752	0.123	3	Mattias Konnebck	male	57	1	2	196.280	S
4	survived	0.932	0.064	0.004	2	Nicole Aguiar Jakobsson	female	23	1	2	130	S
5	lost	0.125	0.752	0.123	2	Oscar Arandes Tejerina	male	31	1	2	91.330	S

Figure 2.2: Prediction for the group members using `gain ratio` in the decision trees in RapidMiner.

2.2 KNIME Insights

This section explains our insights while using KNIME.

2.2.1 Ease of Use

Let us go through the different nodes used to run the decision tree algorithm on our dataset (see Fig. 2.3). We use the *CSV Reader* nodes for both our main dataset (used for training) and our made-up test dataset. It is important to transform the `age` attribute to a float in the test dataset within the *CSV Reader* node to avoid future errors. For the training set, we apply the *Row Filter* node to remove the last row containing missing values. Next, we use the *Rule_Engine* node to combine the information in the `survived` and `body` attributes into a new attribute called `Survival_Result`. For that we write the following code:

$$\$survived\$ = 1 \Rightarrow \text{“Lived”}$$

$$\$survived\$ = 0 \quad \text{AND MISSING} \quad \$body\$ \Rightarrow \text{“Lost”}$$

$$\$survived\$ = 0 \quad \text{AND NOT MISSING} \quad \$body\$ \Rightarrow \text{“Died”}$$

and we select the option *Append Column* and specify the name of our new attribute, i.e., `Survival_Result`. Next, we filter out the non-predictive attributes from both the training and test datasets using the *Column Filter* node.

Before the imputation, we study the age histogram using the *Histogram* and *Image Writer (Port)* nodes, the latter being used to produce a PNG image with the desired size. Once we decide to impute the missing `age` values with the median of their distribution, we use the *Missing Value* node. In the Column Settings, we add the attribute `age` and select the option to impute the missing values with its median.

To run the decision tree algorithm, we use the *Decision Tree Learner* node, where we can configure many parameters, such as the quality measure, choosing between the Gain Ratio or the Gini Index. We then connect our *Decision Tree Learner* node to two other nodes. The first is the *Decision Tree View (JavaScript)*, which provides an interactive option to explore the decision tree. To display the tree, right-click on the node and click on *Open View*. The second node is the *Decision Tree Predictor*, which also takes our preprocessed test dataset as input. This node allows us to obtain the predictions for the test dataset. In this node, we select the option “Change prediction column name” to specify the desired output feature name. We display the survival predictions for our group members in KNIME using the Gain Ratio in Table 2.1.

Name	**Class**	**Sex**	**Age**	**Fare**	**Embarked**	**SibSp**	**Parch**	**Survival_Result**
Anders Dehlbom	2	male	33.0	361.38	Q	2.0	3.0	Lost
Erik Vodopivec Forsman	3	male	23.0	9.35	Q	0.0	0.0	Lost
Mattias Konnebäck	3	male	57.0	196.28	S	1.0	2.0	Lost
Nicole Aguiar Jakobsson	2	female	23.0	130.0	S	1.0	2.0	Lived
Oscar Arandes Tejerina	2	male	31.0	91.33	S	1.0	2.0	Lost

Table 2.1: Prediction for the group members using `Gain_Ratio` in the decision trees in KNIME.

2.2.2 Efficiency

For the Decision Tree algorithm, both approaches appeared to be very efficient. However, the imputation process was much clearer and easier to follow in KNIME.

2.2.3 Costs

An important advantage of Knime is that it is open-source and freely available, making it accessible for both academic research and professional use. Since no license key is required, it can be installed on multiple computers, unlike RapidMiner, where each license is tied to a specific computer.

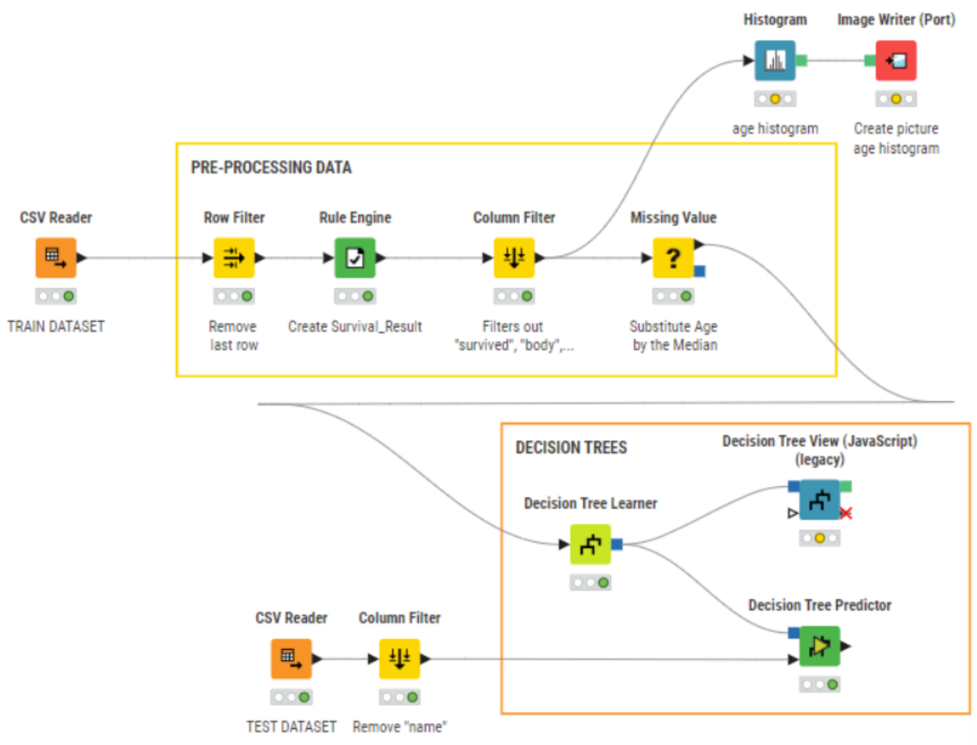


Figure 2.3: KNIME process for the decision tree algorithm.

Chapter 3

Answers to Assignment Questions

3.1 Run your model using gain_ratio. Report your tree nodes, and discuss whether you and the people you know would have lived, died or been lost.

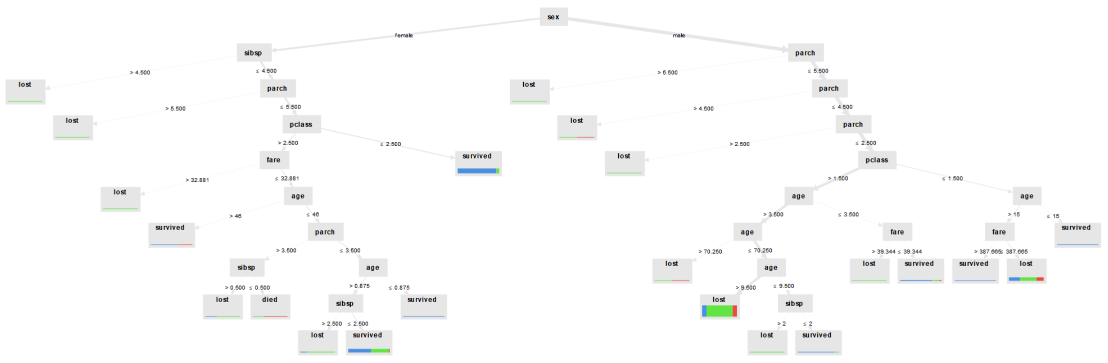


Figure 3.1: Gain ratio decision tree graph.

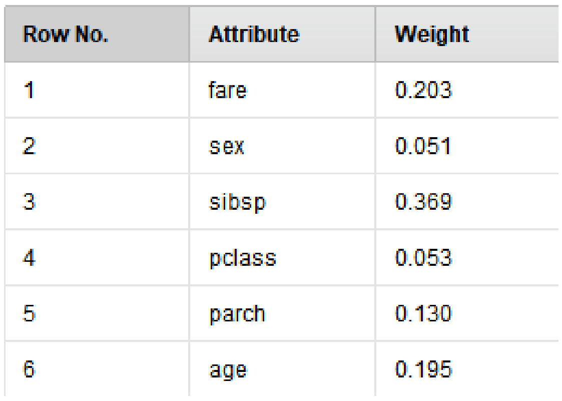
When running the gain-ratio, the first root split is on *sex*, female or male. From the first split, females are directed to family size checks (sibsp and then parch), and after a class check, in which the model refines with fare, age, and smaller sibsp/parch thresholds. Almost all the women end up in the strong survived leaves, except the women in third class. Big families are also classified as lost in the tree as moderate family size tends to survive, and large family size tends to be lost.

- sibsp $> 4.5 \rightarrow$ lost
- else parch $> 5.5 \rightarrow$ lost
- else if pclass $\leq 2.5 \rightarrow$ survived (very strong leaf)
- else (pclass > 2.5) (here it is where the model refines with fare, age, and smaller family sizes, sibsp/parch).

After the sex splitting, men do not go through the same branches as women, they go through:

- parch > 5.5 (or 4.5 / 2.5) $\rightarrow$ lost
- else if pclass > 1.5 and age $> 9.5 \rightarrow$ large lost leaf
- pclass ≤ 1.5 with very high fare gets a survival leaf.

These splits from the tree exhibit a rule where men in 2nd and 3rd class and older than about 10 are predicted to be lost, with only a few rare escapes for first-class cases with extreme fares.



Row No.	Attribute	Weight
1	fare	0.203
2	sex	0.051
3	sibsp	0.369
4	pclass	0.053
5	parch	0.130
6	age	0.195

Figure 3.2: Gain-ratio weight table.

This tree structure also matches the gain-ratio weights table, where sibsp is classed as the most important feature, followed by fare, age, and then pclass, which we as a group thought would be the most important.

When applying the gain-ratio in our test data, we get the results that in our group, everybody (Anders, Mattias, Erik and Oscar) got a high confidence score of lost, except for Nicole, see fig 2.2. Nicole (female, second class and small family) is predicted to have survived with high confidence (≈0.93). The men are predicted as lost due to them being men in 2nd/3rd class, older than age 9.5. Anders is predicted to be certainly lost (1.00), Erik and Mattias are lost at (0.752), and Oscar (2nd class, age close to Anders) is also lost but not as certainly as Anders, which might be related to Anders having the biggest family size on board in our group. Erik and Mattias also have a slightly lower chance of being lost, compared to Anders, despite being in third class, which is also probably due to the family size, because there is no difference in the confidence of being lost between Oscar, Erik, and Mattias despite being in different classes. Because in this labeling "lost" means not recovered and is practically equivalent to death, we interpret P(death) = P(lost) + P(died); which shows that the men's outcomes is overwhelmingly fatal, while Nicole's chance of survival remains extremely high.

3.2 Re-run your model using gini_index. Report differences in your tree structures. Discuss whether your chances for survival increase under Gini or Gain.

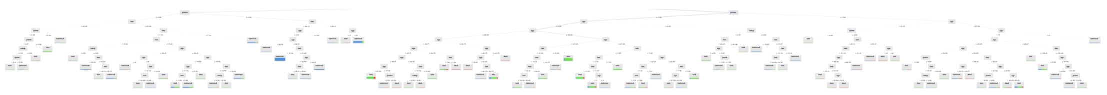
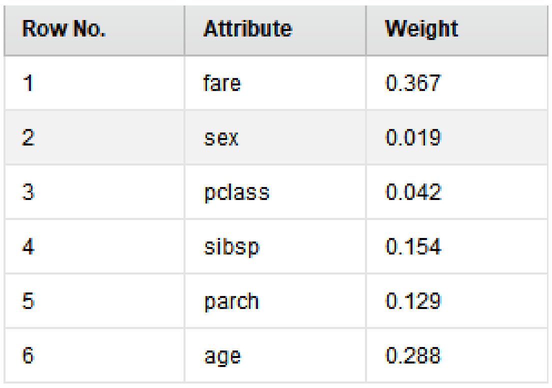


Figure 3.3: Gini index decision tree graph, exhibiting the tree's massive structure.

When switching to Gini index, the tree keeps sex at the root but becomes much more granular inside each branch, gathering long chains of fare and age thresholds (with little pclass in between). The Gini index decision tree is more massive compared to the gain-ratio decision tree, producing several small leaves. Another difference is that the weight distribution is different from the gain ratio. Here, fare and age have a higher weight, whereas in the gain ratio, it was sibsp and then fare.



Row No.	Attribute	Weight
1	fare	0.367
2	sex	0.019
3	pclass	0.042
4	sibsp	0.154
5	parch	0.129
6	age	0.288

Figure 3.4: Gini index weight table

Other differences between Gini index and the gain ratio are that while for females pclass ≤ 2.5 still mostly gives survived, it also carves fine regions with alternating survived/lost based on fare, age, and small family-size changes. For males there are long branches of age and fare splits, where most leaves are still lost/dead, but there are also more narrow survival pockets than in the gain ratio tree.

The group's prediction with this model is that Nicole survives again, but now with even higher confidence (0.993). Anders' leaf is now split between lost = 0.684, died = 0.263, and survived = 0.053, where before it was only certainly lost. Erik and Mattias remain lost with higher confidence than Anders, which might be due to their 3rd-class status, where they paid a low fare (so due to their fare). The interesting change is Oscar: under gini he falls into one of those small fare/age survival pockets for 2nd-class men and is predicted survived (though not very confident with only 0.667). Overall, gini very slightly improves our group's chances of survival because it finds that survival pocket for Oscar, but it does so by carving many fine splits that might risk overfitting. For interpretation, the gain ratio gives a simpler and more stable tree with clear rules.

Chapter 4

Conclusion

In this assignment, we developed a decision tree model to analyze survival patterns from the Titanic disaster. As usual, the original dataset required quite extensive preprocessing, mostly in the way we had to handle missing data, but also in the design of the label attribute, `Survival_Result`. On the other hand, feature selection proved straightforward, as some attributes clearly had a stronger impact than others. The tools used, KNIME and RapidMiner, are great tools, both in the preprocessing of the data and in the building of the model for the algorithm.

The results show that `gender`, `passenger class`, `age`, and `family size` were the most influential attributes affecting survival outcomes. Females and passengers in higher classes had the highest likelihood of survival, while males in lower classes with larger families were most often predicted as lost or deceased. The results therefore quite well align with what is known actually happened in the tragedy of the Titanic: "Women and children first!" and the fact that the ship didn't have a sufficient number of lifeboats, especially not for those on the lower decks.

When comparing the Gain Ratio and Gini Index, both produced realistic outcomes but showed different modeling behaviors. The Gain Ratio created a more compact tree with a better overview, while the Gini Index generated a deeper and more fine-tuned tree that slightly improved the model's sensitivity. As always, there is a trade-off: a more fine-tuned tree may lead to overfitting, simplifying the model by dismissing features or limiting leaf size or tree depth risks missing out on important relationships.

Bibliography

[She00] Colin Shearer. “The CRISP-DM Model: The New Blueprint for Data Mining”. In: *Journal of Data Warehousing* 5.4 (2000), pp. 13–22.

[Sil24] F. Francis Silva. “Techniques for Data Imputation in Data Science: A Technical Review”. In: *Medium* (Oct. 2024). URL: https://medium.com/%40silva.f.francis/techniques-for-data-imputation-in-data-science-a-technical-review-85a626b079dd.