Department of Mathematics and Computer Science
Uncertainty in Artificial Intelligence

Robustness of Classical and Credal Classifiers

Master Thesis
Data Science and Artificial Intelligence

Rob Janssen

Supervisors:
Dr. ir. Erik Quaeghebeur

Other committee members:
David Montalvan Hernandez, MSc
Dr. rer. nat. Noela Müller

20-07-2023

Acknowledgements

First off, I would like to thank the people at Atlas 2 who would listen to me rant every time Python would not do what I wanted it to do, or when writing did not go by itself. Sitting there every day with people who were struggling just like me, made writing a thesis a bit more bearable.

Next, I would like to thank Gennaro Gala and David Montalvan Hernandez. Mainly for helping me better understand how (credal) sum-product networks work and are generated, but also for the feedback they gave me during my midterm presentation. This helped me in adjusting my scope to writing and presenting my work in a more understandable way.

Lastly, and probably most importantly, I would like to thank my supervisor Erik Quaeghebeur. I could always send a message and expect an answer in no time. When having meetings, all attention was on my thesis and questions were always answered clearly and with enough depth. After hearing all kinds of horror stories from fellow students, I think I had a lot of luck with Erik as supervisor. Thank you for all the help during my graduation project.

Abstract

Credal classifiers are, supposedly, more robust classifiers than their classical counterparts, especially when concerned with uncertain data. Credal classifiers are said to be more robust through their conservativeness. However, this definition of robustness has not been tested extensively in the literature and can be better defined.

This thesis attempts to better define the robustness of credal classifiers in the context of missing data. It attempts to quantify this robustness by experimenting with different data sets that contain missing data. Three types of classifiers with classical and credal variants are experimented on, namely naive, tree-based, and sum-product network classifiers. Experiments were run over data sets with generated and inherent missing data.

Credal classifiers, in the context of missing data, seem to perform more robust classifications than their classical counterparts. This mainly holds for the probabilistic classifiers, and to some extent for the tree classifiers. More classifiers could benefit from adding the theory of credal sets to it. This makes classifiers more robust and can make them handle uncertainty in classifications in a better way.

Contents

Chapter 1

Introduction

Models are becoming more prevalent than ever to help with decision-making in all kinds of domains. For instance, in the domain of radiology, classification models are being used to help find cancers in images [9]. These models are used to help in making better, more informed, decisions regarding a patient's health. This is done by showing a classification model patient data and letting it predict whether the patient might have a disease or not. However, this increased use of non-human helpers comes with new problems. What if the models do not make proper classifications? What if models are targeted by adversaries to make bad decisions? According to the European Union, these problems can be resolved by making models more robust, especially those used in critical infrastructures [14].

One type of classifier that is, presumably, more robust than classical classifiers is the credal classifier [22]. Credal classifiers are a type of classifier that make use of credal sets, a set of probability measures, to either create classifiers or make predictions from credal sets. These credal sets either add uncertainty to or represent uncertainty in the credal classifier. Because of these credal sets, credal classifiers can make set classifications instead of just single class classifications. This results in more conservative classifiers, hence making them more robust [20]. In the case of classifying cancers, this can result in multiple types of cancers being classified. This in turn results in a doctor doing further tests to test for both types of cancer, resulting in a more robust prediction of the type of cancer.

One of the first credal classifiers was the naive credal classifier, based on the naive Bayesian classier [39]. This classifier makes use of the imprecise Dirichlet model to add uncertainty to the model outcome, resulting in the creation of credal sets which can be used to make predictions. This naive credal classifier is used as a basis for classification in many newer credal classifiers like credal sum-product networks [17]. Next to these probabilistic

credal models, tree classifiers have also been made credal by using the theory of credal sets in the generation of the tree [20].

These classifiers are said to be more robust, due to the conservativeness in credal classification. However, this robustness is not further defined in any of the papers on credal classifiers. Furthermore, the only paper that attempts to quantify this robustness is the paper by Mauá et al. [22]. Most other papers mention the potential robustness of credal classifiers from a theoretical point of view [20, 39]. To better argue the robustness of some credal classifiers, more concrete evidence could be given.

Given some context and assumptions, credal classifiers will most likely be more robust, as they tend to take uncertainty into account. However, robustness will have to be properly defined and context will have to be given. One type of uncertainty that can have an adversarial effect to classification is the existence of missing data. This thesis will focus on the robustness of classifications, using classifiers that are trained on data sets containing missing data. All raw results and code regarding this thesis can be found on GitHub [16].

Sections 2 and 3 will give a literature review. Section 2 will sketch a timeline of the developments regarding (credal) classifiers and their evaluations, whereas Section 3 will focus more on the mathematical background of some of the theories and classifiers that are used in this report, along with defining the research questions that will be answered in this report. Section 4 will explain the experiments that are done to answer the research questions of this thesis. Answers to the research questions will be given in Section 5 and will be summarized in Section 6. Finally, Section 7 will discuss some limitations of this work, as well as give some recommendations for future work.

Chapter 2

Domain Overview

This chapter will give a conceptual and historical overview of the domain, whilst Chapter 3 will go more in-depth regarding the mathematics behind the models used. First, an overview of some basic concepts regarding classification and credal sets will be given. These will then be used to give more context regarding credal classification and (credal) classifiers. Next, some evaluation measures for (credal) classifiers will be given in the form of accuracy and robustness in different domains. Lastly, a problem formulation will be given with a concrete research question and sub-questions.

2.1 Basic concepts

2.1.1 Classification

What is classification? Classification is a multivariate technique concerned with allocating new objects to previously defined groups, i.e. classes, based on observations on several characteristics of the objects [39]. This definition is very general. Therefore, problems from various domains can be translated into classification problems.

To solve these classification problems, a classifier is usually used. A classifier is a function that maps instances of a set of variables, called attributes or features, to a class, a state of a categorical variable. A classifier usually has to be learned from a data set to be able to classify instances. This is done by taking a sample of some data and letting the classifier learn from this sample. This produces a classifier that can classify similar data. This thesis will focus on two specific types of classifiers: probabilistic classifiers and decision trees.

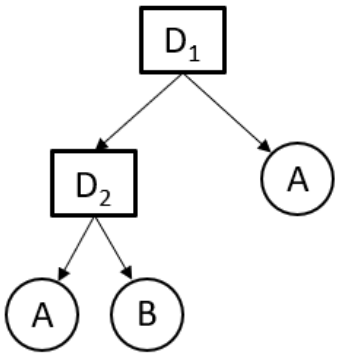

Figure 2.1: Example of a decision tree

Probabilistic classification

Probabilistic classifiers are classifiers that make predictions by choosing the class with the maximum probability [12]. Take a classifier that has been trained to make predictions on some medical data to predict whether someone has disease A or B. This trained classifier is then given an instance, some medical data, for which the doctor wants to know what disease to classify. The classifier gives disease A a probability of $\frac{3}{5}$ and disease B a probability of $\frac{2}{5}$. With these probabilities, the classifier concludes that disease A is the predicted disease, as $\frac{3}{5}$ is greater than $\frac{2}{5}$.

Tree-based classification

Decision trees come to predictions differently. A decision tree is a rooted graph that can be traversed to come to a prediction. This tree is built by creating subsets of data by using a decision rule or strategy until a class can confidently be predicted [30]. When the tree is finished, it can no longer be modified and predictions can be made using it.

For example, take the decision tree in Figure 2.1. The squares denote decision nodes, whilst the circles denote leaves or final predictions. The root of the tree is D_1. Say that this tree is also made to classify whether someone has disease A or B. Starting from the root a path has to be made down to a leaf. D_1 could ask whether there is a history in the family of having disease B. If so, then the arrow, or edge, will be followed to D_2. Here another decision has to be made. Suppose that D_2 has a threshold for blood work. If the value is lower than 2, then the edge to the left is followed to predict the disease to be disease A. Otherwise, the edge to the right is followed to

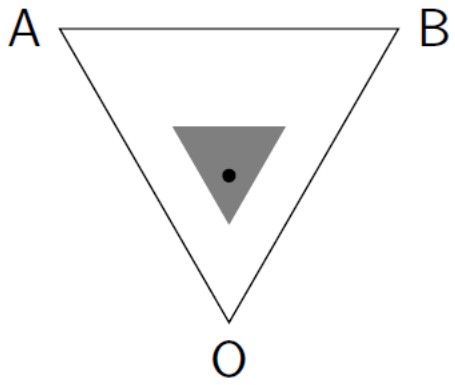

Figure 2.2: Example representation of a credal set [25]

predict disease B.

2.1.2 Credal Sets

A credal set over a categorical variable can be defined as a closed convex set of *probability mass functions* (pmfs) over this categorical variable. An extreme point of a credal set is defined as an element of the set over the categorical variable which cannot be expressed as a convex combination of other elements [3]. Credal classifiers have made use of this representation to handle uncertainty and generate credal sets from which classifications can be made.

Representation

To make this definition a bit more intuitive, take the credal set representation in Figure 2.2. The triangle has 3 corners: A, B, and O. Any point within the triangle represents a pmf. A pmf is a function that gives a probability to a class. In the case of Figure 2.2, the black dot in the middle represents a single pmf, meaning that each class is assigned the same probability. These probabilities could then be used for classical classification. If you were to take the grey triangle, then you would have a set of potential pmfs. Both this black dot, as well as the grey triangle, can be seen as a credal set.

Credal classification

Some classifiers can make classifications based on credal sets [22, 39]. These classifiers are called credal classifiers. Credal classifiers take credal sets and

look for *dominant* classes. All classes that are not being dominated by another class are dominant classes. When predicting, all dominant classes will be classified. This can result in either one unique class or a set of classes being predicted for an instance.

There are two ways in which dominant classes can be found, namely through *interval dominance* and *credal dominance* [39]. Interval dominance takes the lower and upper probability of each class in the credal set and bases the dominance on this. Credal dominance looks at every distribution in a credal set to see whether a class dominates another or not.

Imprecise Dirichlet model

Walley describes an approach for creating credal sets by creating probability intervals, instead of singular probabilities, from data [37]. The paper describes a statistical model called the *imprecise Dirichlet model* (IDM). This model forms the basis for many of the papers that follow in the domain of credal classification. It makes use of Dirichlet distributions to create probability intervals, by finding the lower and upper probability of a set of Dirichlet distributions, where the width of the interval is determined by a hyperparameter s. This means that the model can turn classical classifiers into credal classifiers.

2.2 Classifiers

2.2.1 Naive classifiers

Naive Bayesian classifier

A well-known classical classifier is the naive Bayesian classifier (NBC) [28]. Bayesian classifiers use the Bayes theorem to give each a probability. From these probabilities, the most probable class is predicted per instance. The classifier is called *naive* as it makes the fairly unrealistic assumption that all features are stochastically independent of each other. Even though this classifier is naive and simplistic, it is surprisingly accurate in quite some domains.

Naive credal classifier

An early credal classifier that uses the IDM is the *naive credal classifier* (NCC) [39], which is a credal extension of the NBC. This paper is the basis for many state-of-the-art credal classifiers [22]. This paper was also the first

to introduce interval dominance and credal dominance for the use of credal classification.

Zaffalon also applied the NCC on a dataset containing environmental data [40]. He shows that the NBC is overconfident when classifying instances where the NCC classifies more than one class. When multiple classes are given by the NCC it suggests that there is too much uncertainty to make a unique classification. In the instances where NCC returns multiple classes, it is shown that randomly guessing from this set of classes is better than following the prediction by the NBC.

2.2.2 Extensions of Naive Classifiers

The classifiers from the previous section were created over 20 years ago. In more recent years, these "basic" classifiers have been used as a basis for all kinds of new classifiers. These new classifiers tend to be more complex, but have at least equal, if not better, performance than the classifier they build upon.

Tree-augmented naive credal classifier

Another classifier that uses the IDM as a basis is the *tree-augmented naive credal classifier* (TANC) [41]. A TANC can be seen as a credal network [8] which can be used for credal classification. Classification from the TANC can be done using credal dominance, as suggested for the NCC [39]. It is based on the more classical tree-augmented naive Bayesian classifier [11].

Lazy naive credal classifier

One such classifier is the *lazy* or *local NCC* (LNCC) [7]. LNCC is built on the same principle as the NCC but takes a more local representation of the training set to come to a prediction. The parameter k determines the size of the set of local instances that are used for instance classification. Meaning, the k "closest" instances to the one that is being classified will be used for instance classification. This k is only used as a minimum and a bandwidth selector will tune the amount of "close" instances until the instance is classified, or all instances in the training set are used to make an instance classification.

Next to the LNCC the paper by Corani and Zaffalon also proposes a way of comparing credal classifiers based on a *discounted accuracy* [7]. When a set is predicted for an instance and it contains the correct class, then it is seen as partially correct. This method of comparison is used to evaluate NCC to

the suggested LNCC. It shows that the LNCC can significantly outperform the NCC. However, this is not always the case and in one case the NCC even significantly outperforms the LNCC.

Ensemble methods

Ensemble methods have also been used in the credal domain. Either by making an ensemble over credal classifiers [1, 23], or by creating a credal ensemble over Bayesian classifiers [6]. The ensembles of credal classifiers seem to be able to classify instances with unique classes more often than their singular classifiers [23] and they outperform ensemble methods with non-credal classifiers. The credal ensemble of Bayesian classifiers seems to also outperform traditional ensemble methods which do not make use of credal sets [6].

Probabilistic decision graph

In recent years, probabilistic classifiers in the form of networks or graphs have become more prevalent. One such classifier is the probabilistic decision graph (PDG) [15]. PDGs are a way of representing probability distributions through a graphical model. Using this representation, probabilistic inferences can be made.

Sum-product network

Another type of network is the *sum-product network* (SPN) will be discussed [24]. This classifier makes use of sum and product nodes to come to classifications. This can be done by either finding the probabilities for each class or by doing a backward pass through the network to find the optimal class for an instance. One way of creating the structure of an SPN is through an algorithm called LearnSPN [13].

Credal sum-product network

This SPN has been made into a *credal sum-product network* (CSPN) by contaminating the sum-nodes [22]. This classifier uses sum and product nodes to generate probability intervals for a given input, instead of singular probabilities. This interval comes from the interval weights used on the edges of sum nodes. This output can then be used for classification purposes. The CSPN was created to determine whether classifications of a normal SPN are reliable or unreliable. This was done by measuring the robustness of instances, which is discussed in more detail in Section 2.3.3.

Another way of creating a CSPN is through the use of the algorithm called LearnCSPN [17]. Instead of contaminating, or building on, an existing structure it creates a different structure from the data. LearnCSPN, in comparison to LearnSPN, is capable of creating a structure over data that contains missing data.

2.2.3 Tree-based classifiers

C4.5 algorithm

Another improvement on naive classifiers is the decision tree built by the *C4.5* algorithm [29]. This algorithm is still one of the most used algorithms for building decision trees. The algorithm iteratively looks at the attributes of the given training data to generate a decision tree based on the best attribute to split on. This is done by finding the maximum value given by the split criterion. The tree is generated by splitting the data into smaller subsets until all instances share the same unique class. This results in a leaf for this class. To make predictions the decision tree can be iterated over with a new instance. This instance will find a path down to a leaf, which will result in a prediction for this leaf.

Credal-C4.5 algorithm

C4.5 also has a credal variant, namely *credal-C4.5* [19]. This algorithm has been adapted to the credal domain, with the main difference being a different split criterion for creating a decision node. This comes from the use of IDM in the credal split criterion, resulting in added uncertainty. This results in different attributes being selected to split the data on, which in turn results in a different and arguably more robust decision tree than the one from C4.5. The paper by Mantas and Abellán shows that credal-C4.5 performs similarly to C4.5 when there is no missing data, whilst outperforming these classifiers when missing data is added to a data set [19].

In a follow-up paper, Mantas et al. evaluate C4.5 and credal-C4.5 on different data sets, with differing amounts of noise added to the data [20]. They also experiment with differing levels for s in credal-C4.5, which adjusts the width of the probability interval behind the credal set through the IDM. It shows the same results as the original paper for C4.5 compared to credal-C4.5 [19]. It also suggests that optimizing s for improving classification is positive.

2.3 Evaluating (credal) classifiers

2.3.1 Accuracy

Prediction accuracy

Prediction accuracy has been the most prevalent standard for evaluating classification models [33]. Assessing models on the metric of prediction accuracy lets one conclude that the more often a classifier makes correct predictions, the better the classifier is. Almost all classifiers in any domain have been optimized on this metric. As such, optimizations of models have led to highly accurate models. However, these models tend to become less robust. Su et al. show this problem in the domain of image classification models, which have become more accurate and less robust over time [33]. This entails that classifiers are performing worse when "odd" instances have to be predicted by the classifier.

Trade-off

This trade-off between accuracy and robustness has had a lot more attention in academic literature in recent years. Whether it is truly a trade-off is also something that is still being debated in current literature. Yang et al. suggest that accuracy and robustness can both be achieved together in a model [38]. They show that the trade-off can exist, but is not always a given. On the other hand, Tsipras et al. suggest that there is an inherent trade-off between standard accuracy and adversarial robustness of a model [36]. Both papers are written in the domain of image classification but suggest opposite views on a possible trade-off between accuracy and robustness. Both papers suggest that not only accuracy is important for the performance of a model. Robustness is a metric that should also be taken into account, and in some domains should be more important than accuracy.

2.3.2 Robustness

Robustness is also becoming more important for policymakers, as is outlined by a report from the European Union [14]. Algorithms and artificial intelligence are implemented in lots of technological systems to help with, for instance, decision-making. These systems that handle decision-making could be targeted by adversaries who are interested in disturbing proper decision-making. As such, the robustness and explainability of systems have become of greater importance over time. Even so much so that the European Union

is now actively making policies for businesses to make their technologies more robust.

Defining Robustness

A dictionary would define robustness as: "capable of performing without failure under a wide range of conditions" [35]. This definition, whilst being a solid overall definition, is rather vague. The rest of this Section will focus on more precisely defining robustness in the domain of classification.

One way of defining robustness in classification is the strength of the attack that is needed for a system to misclassify an instance that was previously not being misclassified by the system. The stronger an attack is needed to make a system make an error, the more robust the system is against these types of attacks. These adversarial attacks are still used to evaluate the robustness of state-of-the-art neural networks [2, 4].

Good practices for evaluating robustness

Carlini et al. have attempted to create an overview of the foundations, good practices, and new methods of evaluating adversarial attacks [5]. The paper is written as a framework, but should not be used as one. The authors explicitly state that any researcher should use their knowledge and expertise to judge what practices are needed in their research, as some recommendations might be inapplicable to some domains. For instance, this thesis will not focus on adding missing data until a classifier can no longer classify. Even though not all recommendations will be applied in this report, the paper does contain good practices on how robustness should be evaluated, using examples of what went well and what did not in the past. This paper will be used to guide the eventual evaluation criteria for robustness.

Adversarial attacks

One problem with adversarial attacks is that these are usually synthetic. These attacks can recreate what adversarial persons could do to a system to let it malfunction. However, there also exist natural disturbances in data that can result in misclassifications. According to Taori et al., measures taken to become more robust to synthetic disturbances have little effect on robustness towards natural disturbances [34]. They also suggest that it might be of more importance for a model to be robust to natural disturbances in data, as these can appear in real-life data. The goal of classifiers in practice is to be able to classify beyond their training and test data and to be able to classify real-life data. As such, it should, in most cases, be more important for robust models

to be able to classify naturally disturbed data over synthetically disturbed data. This can be seen as a trade-off dependent on the context. In cases where adversaries want to attack a system that uses a classification model then it would be more important that the model is robust against all forms of adversarial attacks.

Taori et al. look at robustness in ImageNets [34]. They measure the difference in accuracy between the test set and a naturally shifted test set. Another type of natural disturbance can be the occurrence of missing data. Missing data adds a layer of uncertainty to classifications. Most classical models do not function on incomplete data and need imputation methods to function. However, there also exist models that can handle uncertainty in the form of missing data.

Robustness of credal classifiers

Credal classifiers are said to be more conservative classifiers than their classical variants, from which is concluded that credal classifiers are more robust [21, 22]. Conservativeness, in this context, means that the classifier is less prone to make conclusive, unique, class predictions. This conclusion of being a more robust classifier is drawn under the assumption that it is better, for a classifier to predict a set of classes when it is not fully certain rather than making a potential misclassification.

2.3.3 Evaluating classifiers

Evaluating accuracy

Stapor states that accuracy (or classification error) has been the most prevalent metric for evaluating classifiers for years [32]. However, there exist a lot more metrics to evaluate classifiers, for instance, recall, specificity, and precision. All the given metrics have their specific use cases. What metric is used, is usually up to a researcher and what they want to find. All the given metrics state something about the performance of a classifier, under certain assumptions.

Mantas et al. make a comparison of a credal classifier and its non-credal variant [20]. They evaluate the differences in performance on all kinds of data sets with differing amounts of noise over the data. Using this method, they can make a clear distinction between the classifiers, and can even show how the parameter of the credal classifier can have an impact on the results. This paper only argues about the difference in accuracy. However, the results of the paper could be extended to suggest whether a classifier is more robust

than another.

Evaluating robustness

The paper by Mauá et al. on classification using SPNs specifically focuses on evaluating the robustness of instances by using a credal classifier [22]. Robustness is measured by the amount of contamination that is needed to lead to either set classification or misclassification. The more contamination is needed to result in a set classification or misclassification, the more robust an instance can be classified. This robustness is then normalized and compared to the accuracy of the normal SPN. This shows that robust instances are more often classified accurately than less robust instances when noise is added.

2.4 Formulation of the problem and objectives

Credal classifiers are said to be more robust than their classical counterparts. However, this statement has only been supported by the fact that credal classifiers are more conservative. Most credal classifiers have not been evaluated against their original variant on robustness. Most papers either only focus on accuracy or focus on robustness in another way, see Section 2.3.3.

The main research question that will be tackled in the master thesis will be:

When and how, in relation to missing data, are credal classifiers more robust than their non-credal counterparts, and to what degree does this generalize?

This research question will be subdivided into multiple sub-questions:

SQ1 Are credal classifiers more robust than classical classifiers?

SQ2 What is the difference in robustness between credal classifiers and their classical counterpart under varying levels of missing data?

SQ3 Are there similar differences in performance on differing levels of missing data between the models in their classical form, as well as their credal form?

SQ4 Do there exist more classifiers, that have no credal variant yet, that can be made credal for more robust classifications?

Chapter 3

Classifier Overview

This chapter goes into more depth regarding the theoretical background of the theories and models that are used in this thesis. First, some basic concepts will be discussed regarding classification and credal sets. After which the classical and credal models will be explained in more detail, better showing what makes them different.

3.1 Basic concepts

3.1.1 Classification

Probabilistic classification

The formal definition of probabilistic classification that will be used in this thesis comes from Zaffalon [39]: A classification variable can be formally denoted by C, taking values in the finite set $\mathscr{C}$, where the possible classes are denoted by lower-case letters. We measure n features $A_1, ..., A_n$ taking generic values $a_1, ..., a_n$ from the sets $\mathscr{A}_1, ..., \mathscr{A}_n$, which are assumed to be finite. We can define a probabilistic classifier through the discrete joint probability distribution $P[C, A_1, ..., A_n]$. The classification of a new pattern $(a_1, ..., a_n)$ is realized by selecting a class $c \in \mathscr{C}$ that maximizes $P[c|a_1, ..., a_n]$.

Tree classification

This definition can only be used for probabilistic classification, as a decision tree makes classifications differently. A decision tree is a graph $G = (N, E)$ consisting of finite nodes (N) and edges (E). More specifically, a decision tree is a directed acyclic graph. Directed means that the edges are ordered pairs (n, m) of vertices. Acyclic means that the graph does not contain cycles.

These restrictions result in the following properties for a directed acyclic graph, and therefore also for a decision tree:

1. There is exactly one node that has no incoming edges, called the root.
2. Every node, apart from the root, has exactly one incoming edge.
3. There exists a unique path to each node, from the root.

As such, a decision tree can be used to classify instances. This is done by starting at the root and finding a unique path to a leaf, a node with no outgoing edges. This leaf will contain a class, which is used to classify instances.

3.1.2 Credal Sets

The formal definition of credal sets that will be used in this master thesis comes from Antonucci et al. [3] and is defined in the following way: A credal set $M(X)$ over a categorical variable X can be defined as a closed convex set of probability mass functions over X. An extreme point of a credal set is an element of this set that cannot be expressed as a convex combination of other elements. This thesis will focus on finitely-generated credal sets, meaning sets with a finite number of extreme points.

A credal set can be defined in two different, yet formally the same, ways. Either as a convex hull of a set of distributions or through linear constraints. In both cases, the probability of a generic event lies within an interval whose extremes are the minimum and maximum of the probability of when the distribution varies in the credal set. These are also called the lower ($\underline{P}$) and upper probability ($\overline{P}$).

Representations

In Figure 3.1 two representations of credal sets are shown. The left one shows a singleton credal set. This credal set contains a single probability mass function (pmf), namely $p = (p_A, p_B, p_O) = (\frac{1}{4}, \frac{1}{4}, \frac{1}{2})$. As this credal set contains only one pmf, it can also be used by a classical probabilistic classifier to make a classification. In this case, p_O is the greatest, meaning class O will be predicted. This singleton credal set can be turned into a linear-vacuous credal set, by using the IDM.

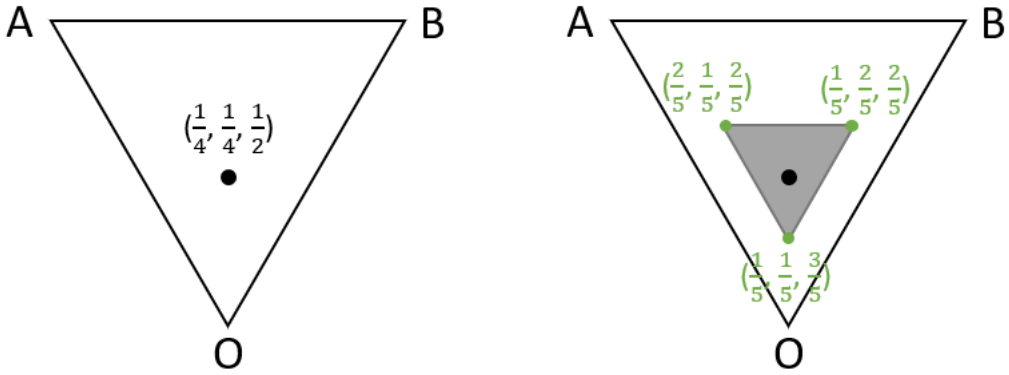

Figure 3.1: Two credal set representations

Imprecise Dirichlet model

The imprecise Dirichlet model (IDM) can be used to add uncertainty to the probability of an instance [37]. This results in probabilities changing from single values to intervals, which are derived from a linear-vacuous credal set. The equation for creating the lower ($\underline{P}$) and upper ($\overline{P}$) bound for the probabilities are:

$$\underline{P}(y) = \frac{\text{freq}(y)}{|T| + s} \tag{3.1}$$

and

$$\overline{P}(y) = \frac{\text{freq}(y) + s}{|T| + s} \tag{3.2}$$

where freq(y) gives the number of times an event y is observed. $|T|$ is the total number of observations or instances, and s is a hyperparameter that can be defined as the number of "hidden" observations.

Take the singleton credal set from Figure 3.1. Suppose that the singleton credal set comes from a data set with 8 instances. For this example, s will be set to 2. To create a linear-vacuous credal set, the extreme points of the credal set will have to be found. Taking the given information and the formulas for $\overline{P}$, it is possible to calculate the upper probability of class O. Given $p_O = \frac{1}{2}$ and 8 total observations, class O given some event was observed 4 times. So freq(O) $= 4$, $N = 8$, and $s = 2$, gives $\overline{P}(O) = \frac{3}{5}$ by using Equation 3.2. Together with $\underline{P}(A)$ and $\underline{P}(B)$, the extreme point $p(\frac{1}{5}, \frac{1}{5}, \frac{3}{5})$ can be found. By finding all extreme points the credal set on the right of Figure 3.1 can be generated.

3.1.3 Credal Classification

From credal sets, it is possible to make classifications using so-called credal classification. Credal classification makes use of the dominance of classes to determine how to classify an instance. This dominance is found by looking at interval dominance and credal dominance in credal sets.

Interval dominance

Interval dominance looks, as the name suggests, at intervals. It takes the lower and upper probability of each class in the credal set and finds what unique class or set of classes is undominated.

Definition of interval dominance:

Definition 1 (Def. 3.1. of Zaffalon [39])**.** *Let X be a discrete random variable defined over $\mathscr{X}$ and let $\mathscr{X}', \mathscr{X}'' \subseteq \mathscr{X}$ be two generic events. Let A represent what is known, and let the probabilities $P[\mathscr{X}'|A]$ and $P[\mathscr{X}''|A]$ be, respectively, represented by the intervals $I' = [\underline{P}[\mathscr{X}'|A], \overline{P}[\mathscr{X}'|A]]$ and $I'' = [\underline{P}[\mathscr{X}''|A], \overline{P}[\mathscr{X}''|A]]$. The interval I' is said to dominate I'' if $\underline{P}[\mathscr{X}'|A] > \overline{P}[\mathscr{X}''|A]$; in this case $\mathscr{X}'$ is said to interval dominate $\mathscr{X}''$.*

The idea behind this definition is that since each probability in I' is greater than each probability in I'', $\mathscr{X}'$ is always more probable than $\mathscr{X}''$. As such, the class with I' will be classified.

Credal dominance

Next to interval dominance there also exists credal dominance. This way of finding dominance is not used for classification purposes in this thesis, however, for the sake of completeness it will still be explained.

Credal dominance takes a more in-depth look at a credal set than interval dominance does. In essence, it takes each possible pmf in the credal set, resulting in more tests and more possibilities for finding dominance relations.

Definition of credal dominance:

Definition 2. *(Def. 4.1. of Zaffalon [39]) Let X be a discrete random variable defined over $\mathscr{X}$ and let $\mathscr{X}', \mathscr{X}'' \subseteq \mathscr{X}$ be two generic events. Consider the distribution $P[X|A] \in \mathscr{P}_X^A$, where A represents what is known and $\mathscr{P}_X^A$ is a non-empty set of distributions. $\mathscr{X}'$ is said to be credal dominant as compared to $\mathscr{X}''$, $\mathscr{X}' \succ \mathscr{X}''$, if for every distribution $P[X|A] \in \mathscr{P}_X^A$, $P[X'|A] > P[X''|A]$.*

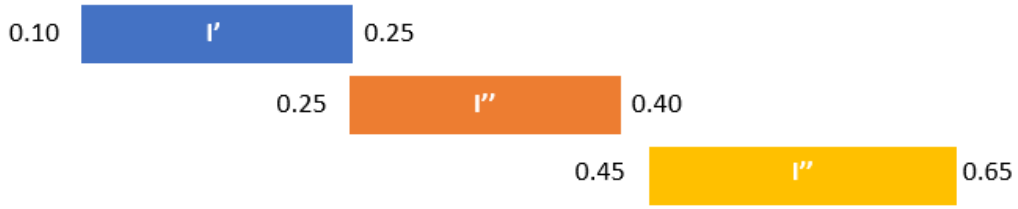

Figure 3.2: Visualisation of interval dominance

The intuition in credal dominance is that, given some probability distributions, if class c' is more probable than class c'' in all distributions, then class c' is dominant over class c''. Credal dominance is a strengthening of interval dominance, it says more than interval dominance does. Interval dominance implies credal dominance, whilst the reverse does not have to be true. When the intervals do not overlap, then the smallest value of the dominant interval is always greater than the highest value of the dominated interval, which can also be used to conclude credal dominance.

Example

To give some more intuition, take the following example. Suppose that $\mathscr{C} = \{c', c'', c'''\}$ and that $\mathscr{P}_C^{a_1,...,a_n}$, denoting a set of distributions $P[C|a_1, ..., a_n]$, has three pmfs, $p = (p(c'), p(c''), p(c'''))$: $p_1(0.10, 0.25, 0.65)$, $p_2(0.15, 0.40, 0.45)$, and $p_3(0.25, 0.30, 0.45)$, where the elements of the vectors are $P[c'|a_1, ..., a_n]$, $P[c''|a_1, ..., a_n]$, and $P[c'''|a_1, ..., a_n]$, respectively. The intervals that these posterior probabilities belong to are $I' = [0.10, 0.25]$, $I'' = [0.25, 0.40]$, and $I''' = [0.45, 0.65]$, respectively.

In this example, you can see from the numbers and Figure 3.2 that c''' interval dominates c' and c'', so c''' will be classified. However, the intervals for c' and c'' overlap, so there exists no interval dominance between these two. When looking at credal dominance, there is dominance to be found between c' and c''. In each separate pmf, c'' is greater than c', as also depicted in Figure 3.3. As such, c'' credally dominates c'.

3.2 Naive classifiers

3.2.1 Naive Bayesian classifier

The naive Bayesian classifier (NBC) takes the definition of probabilistic classification and adds the assumption of independence of all features, conditional

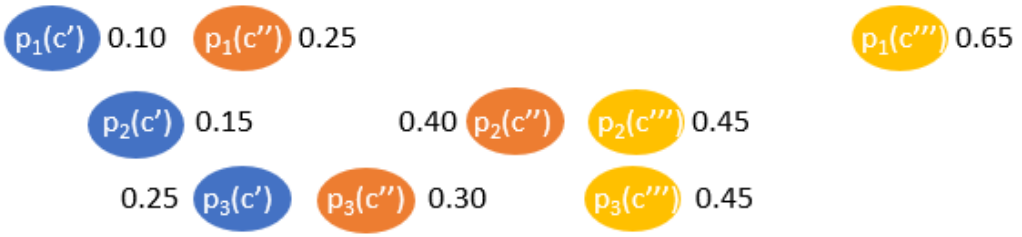

Figure 3.3: Visualisation of credal dominance

on the class:

$$P[A_1, ..., A_n|C] = \prod_{i=1}^{n} P[A_i|C] \tag{3.3}$$

Without this assumption, an exponentially growing number of probabilities, in terms of the number of attributes, would be needed to define the joint distribution [39]. Adding this assumption makes calculating Bayes theorem, Equation 3.4, more doable.

$$P[C|A] = \frac{P[A|C]P[C]}{P[A]} \tag{3.4}$$

3.2.2 Naive credal classifier

The NBC can then be extended to a naive credal classifier (NCC) using the theory of credal sets. By taking Equation 3.3 as a basis and by associating a credal set to each random variable, it is possible to define the NCC. Let $\mathscr{P}_C$ denote a set of probability mass functions $P[C]$. For a generic attribute A_i and for each $c \in \mathscr{C}$, let $\mathscr{P}^c_{A_i}$ denote a credal set of the conditional distributions $P[A_i|c]$. These sets can be referred to as local credal sets.

Definition 3. *(Def. 2.1. of Zaffalon [39]) The NCC is the model characterized by the set $\mathscr{P}$ of joint distributions $P[C, A_1, ..., A_n]$ that are obtained by assuming Equation 3.3 and making every possible combination of the distributions in the local credal sets,*

$$\mathscr{P} = \left\{ P[C] \prod_{i=1}^{n} P[A_i|C] : P[C] \in \mathscr{P}_C, P[A_j|c] \in \mathscr{P}^c_{A_j}, j = 1, ..., n, c \in \mathscr{C} \right\}. \tag{3.5}$$

Definition 3 of the NCC by Zaffalon shows that local credal sets are all that is needed to build the classifier. Now that credal sets are generated, interval dominance and credal dominance can be used to make inferences using the NCC.

A_1	A_2	C
0	0	A
0	1	A
0	1	A
1	1	A
1	1	A
1	1	A
1	1	A
1	1	A
1	1	A
1	1	A
0	0	B
0	0	B
0	0	B
0	0	B
0	0	B
0	1	B

Table 3.1: Example data to illustrate classifiers

3.2.3 Example

To make each classifier less conceptual and more practical, an example will be given of how the classifiers are trained and come to predictions. For each of these examples the data in Table 3.1 will be used to illustrate the training of a classifier.

Take an instance with attributes $(a_1, a_2) = (0, 1)$. The NBC and NCC have both been trained on the example data and want to predict the given instance. Both will first have to calculate the probability for each class given the instance, using Equation 3.4:

$$P(c_{\mathrm{A}}|a_1, a_2) = \frac{\frac{2}{9} \cdot \frac{9}{15}}{\frac{3}{15}} = \frac{2}{3}$$

and

$$P(c_{\mathrm{B}}|a_1, a_2) = \frac{\frac{1}{6} \cdot \frac{6}{15}}{\frac{3}{15}} = \frac{1}{3}$$

The NBC can take these probabilities and conclude that $\frac{2}{3} > \frac{1}{3}$, thus class A is predicted. The NCC will have to run the IDM to make a prediction. For illustration purposes, Equations 3.1 and 3.2 will only be shown for class A. $\mathrm{freq}(c_{\mathrm{A}}|a_1, a_2)$ will be calculated by taking $P(c_{\mathrm{A}}|a_1, a_2)$ and multiplying

it by the total number of instances given (a_1, a_2), and s will be set to 2.

$$\underline{P}(c_{\mathrm{A}}|a_1, a_2) = \frac{\frac{2}{3} \cdot 3}{3+2} = \frac{2}{5}$$

and

$$\overline{P}(c_{\mathrm{A}}|a_1, a_2) = \frac{\frac{2}{3} \cdot 3 + 2}{3+2} = \frac{4}{5}$$

Thus, class A gets the probability interval $[\frac{2}{5}, \frac{4}{5}]$ from $\underline{P}$ and $\overline{P}$. Doing the same for class B results in the probability interval $[\frac{1}{5}, \frac{3}{5}]$. Neither class is interval dominant as the intervals overlap. Therefore the NCC classifies both classes A and B as a set of classes.

3.3 Tree-based classifiers

3.3.1 C4.5 algorithm

The C4.5 algorithm [26] generates decision trees that can be used for classification. This means that one can go from the root of the tree to a leaf by making a path that passes through decision nodes. The path is generated by choosing a path in each decision node that matches the attribute value in the instance that is being classified. The path ends in a leaf where a class is defined, which can be chosen as the best possible class for the given instance. The construction algorithm of the decision tree is defined in Algorithm 1.

Let T be the set of instances. ComputeClassFrequency(T) computes the weighted frequency of cases in T whose class is C_i for $i \in [1, |C|]$. If all cases in T belong to the same class C_j, then the node becomes a leaf associated with class C_j. If T contains cases that belong to multiple classes, then the *information gain* of each attribute is calculated by the function GainCriterion(A). For discrete attributes, the information gain is relative to the splitting of cases in T into sets with distinct attribute values, whereas for continuous attributes the information gain is relative to the splitting of T into two subsets, which is determined during information gain calculation. The attribute with the highest information gain value is selected for evaluation in the decision node. The decision node splits into z nodes, with $z = 2$ for a continuous decision node with a threshold, and z is equal to the number of categories for the discrete decision node. FormTree is called recursively until the whole tree is generated.

The paper by Ruggieri also adds pruning and early stopping to the pseudocode [29]. These are not discussed in this literature review or mentioned in Algorithm 1.

Algorithm 1 FormTree(T) [29]

```
ComputeClassFrequency(T)
if OneClass then
    return a leaf
end if
create a decision node N
for each Attribute A do
    GainCriterion(A)
end for
N.test = AttributeWithBestGain
if N.test is continuous then
    find Threshold
end if
for each T' in the splitting of T do
    if T' is Empty then
        Child of N is a leaf
    else
        Child of N = FormTree(T')
    end if
end for
```

The information gain criterion is the main difference between the standard C4.5 algorithm and the credal variant. The standard variant will make use of the *information gain ratio criterion*. This criterion is defined as the ratio between the information gain and split information. This is better than just taking information gain, as it takes missing data into account [27]. The information gain ratio is defined as:

$$\text{ratio}(T) = \frac{\text{gain}(T)}{\text{split}(T)} \tag{3.6}$$

with information gain being defined as:

$$\text{gain}(T) = \text{info}(T) - \sum_{i=1}^{z} \frac{|T_i|}{|T|} \cdot \text{info}(T_i) \tag{3.7}$$

where

$$\text{info}(T) = -\sum_{j=1}^{NClass} \frac{\text{freq}(C_j, T)}{|T|} \cdot \log_2(\frac{\text{freq}(C_j, T)}{|T|}) \tag{3.8}$$

and with the split criterion being defined as:

$$\text{split}(T) = -\sum_{i=1}^{z} \frac{|T_i|}{|T|} \cdot \log_2(\frac{|T_i|}{|T|}) \tag{3.9}$$

where $i \in [1, z]$, and $|T|$ and $|T_i|$ are the cardinality of T and T_i, respectively.

3.3.2 Credal-C4.5 algorithm

On the other hand, credal-C4.5 assumes some amount of noise in the data and uses *imprecise information gain* to ratio with the split criterion. This is defined as:

$$\text{impreciseGain}(T) = \text{info}(K^D(T)) - \sum_{i=1}^{z} P^D \cdot \text{info}(K^D(T_i)) \tag{3.10}$$

where the function K^D creates a credal set using the IDM, and P^D is a probability distribution that belongs to the credal set $K^D(T_i)$.

The formula K^D iterates over the hyperparameter s of the IDM. It iteratively adds 1 sample, or the remainder of s, to the T_i that is observed the least in T, until s samples are added. This results in the fractions in info(T) coming closer to $\frac{1}{2}$, which maximizes info(T).

3.3.3 Example

To show how the trees are generated from data, the data in Table 3.1 are used again. The data correspond to the data as used in the example by Mantas et al. [20]. The resulting example, however, is not the same, as when calculating all values it was found that the example does not follow the equations in the paper.

To decide on what attribute the tree has to make a decision node, both C4.5 and credal-C4.5 will have to calculate their criteria. C4.5 calculates the information gain ratio as follows:

$$\text{ratio}_{A_1}(T) = \frac{0.971 - 0.551}{0.971} = 0.443$$

and

$$\text{ratio}_{A_2}(T) = \frac{0.971 - 0.562}{0.971} = 0.421$$

whilst credal-C4.5, with $s = 2$, calculates the imprecise information gain ratio as follows:

$$\text{ratio}_{A_1}(T) = \frac{0.998 - 0.933}{0.971} = 0.036$$

and

$$\text{ratio}_{A_2}(T) = \frac{0.998 - 0.910}{0.971} = 0.060$$

For the given data, the algorithms make different trees. C4.5 first splits on A_1, as $\text{ratio}_{A_1}(T) > \text{ratio}_{A_2}(T)$, whilst credal-C4.5 first splits on A_1, as $\text{ratio}_{A_1}(T) < \text{ratio}_{A_2}(T)$.

Only the tree from C4.5 will be used to illustrate how tree classifiers make predictions, as both algorithms make trees that make predictions in the same way. Normally when multiple classes are assigned to the same leaf, the leaf would be pruned as those instances cannot be uniquely classified by the tree. For the sake of this example the class that appears most in the leaf was chosen instead of pruning the leaf. Given this added assumption, C4.5 in the end creates the tree as shown in Figure 3.4. When predicting an instance with attributes $(a_1, a_2) = (0, 1)$, it will go from the root (A_1) to another decision node (A_2) and ending in the leaf where c_A is predicted. Thus, the instance is predicted to belong to class A.

3.4 Sum-product network classifiers

3.4.1 Sum-product network

An SPN is a rooted directed acyclic graph with variables as leaves, sum- and product-nodes as internal nodes, and weighted edges [31]. This type

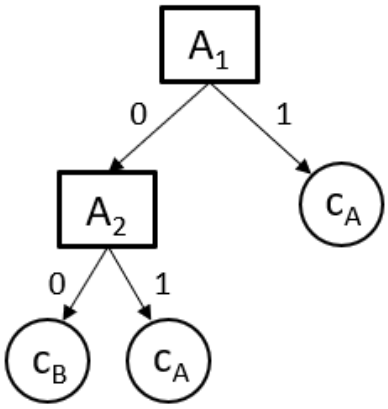

Figure 3.4: Tree created by C4.5 from example data

of network was coined as a way to make the partition function tractable to take away a key limiting factor of graphical model inference and learning. A multitude of algorithms has been created for generating SPNs from data. The one that will be highlighted in this thesis is *LearnSPN*, as a credal variant of this algorithm has also been developed.

LearnSPN

Gens and Domingos first introduced LearnSPN [13]. Algorithm 2 shows the pseudocode for LearnSPN, which is also visually represented in Figure 3.5. LearnSPN is an algorithm schema rather than a single algorithm. A multitude of methods can be used for splitting features (V) and instances (T) into subsets.

Algorithm 2 recursively takes T and A until $|A| = 1$. When only a single feature is left, univariate distributed leaves can be created based on the values of A in T. If multiple features are left, no leaves can be generated, thus a sum or product-node will have to be made. First, A will be partitioned into approximately independent subsets A_j. The method for approximation is something that is not predetermined by LearnSPN. If a partition can be made, then a product-node will be generated for each subset A_j, from which LearnSPN will be called again, with T. If no partition can be made, then first T will be partitioned into similar instances T_i. Again, the method for clustering is something that is not predetermined by LearnSPN. From these partitions, a sum-node will be generated for each subset T_i, from which LearnSPN will be called again, with A and edge weights $\frac{|T_i|}{|T|}$.

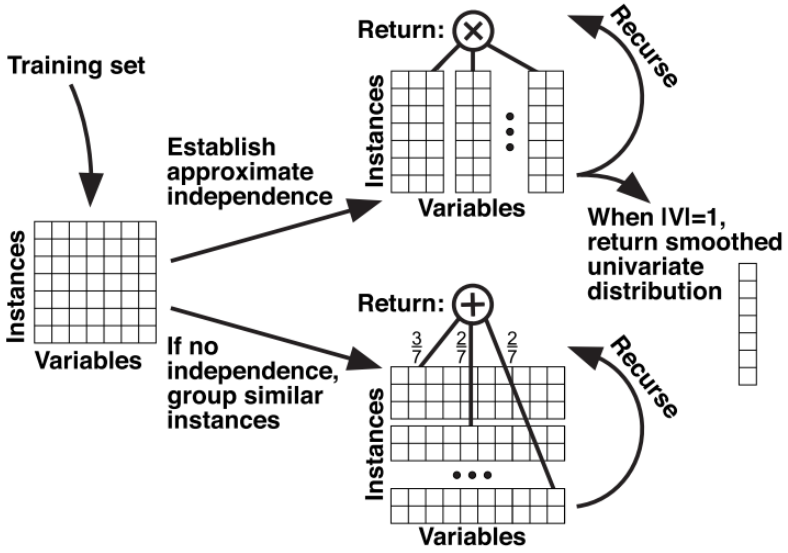

Figure 3.5: Visual representation of LearnSPN [13]

Algorithm 2 LearnSPN(T, A)[13]

Require: set of instances T and set of variables A
Ensure: an SPN representing a distribution over A learned from T
  if $|A| = 1$ **then**
    return univariate distribution estimated from the variable's values in T
  else
    partition A into approximately independent subsets A_j
    if success **then**
      return $\prod_j$ LearnSPN(T, A_j)
    else
      partition T into subsets of similar instances T_i
      return $\sum_i \frac{|T_i|}{|T|} \cdot$ LearnSPN(T_i, A)
    end if
  end if

Creating nodes In this report, a group-wise independence test (G-test) is chosen to test independence and create product-nodes. Expectation maximization (EM), though the Gaussian mixture model (GMM), is chosen for clustering and creating sum-nodes. These are chosen as they are used in the original LearnSPN paper by Gens and Domingos [13]. To create leaves, maximum likelihood estimations are used, as these are implemented in DeeProbkit [18].

G-test G-test compares two attributes to see whether they are independent of each other. This test works best on discrete values. Therefore, continuous attributes are discretized when running a G-test. The lower the outcome of the G-test the more likely it is that two attributes are independent. If, given some predetermined threshold, attributes are found to be independent, then they are split by a product-node.

Formally, the G-test is defined as:

$$G(A_1, A_2) = 2 \sum_{a_1} \sum_{a_2} \text{freq}(a_1, a_2) \cdot \log \frac{\text{freq}(a_1, a_2) \cdot |T|}{\text{freq}(a_1) \cdot \text{freq}(a_2)} \tag{3.11}$$

where A_1 and A_2 are the attributes that are compared for independence and log is the natural logarithm. This equation gives a G-value that can be used to determine whether A_1 and A_2 are independent of each other. The threshold is set to 5 in this thesis, meaning that if the G-value is below 5 then the attributes are seen as independent and a product-node will be made.

EM-algorithm When no independence can be found between any of the attributes, then instances will have to be clustered. The instances will always be clustered in two clusters. Clusters are initialized using K-means, after which EM is run using the GMM to find the optimal parameters for the underlying distribution. These clusters are then split by a sum-node. Each edge is given a weight $\frac{|T_i|}{|T|}$.

The EM-algorithm maximizes the log-likelihood of each cluster:

$$\log(t) = \sum_{var} \log((w_{val} + smoo)/(|T| + 2 \cdot smoo)) \tag{3.12}$$

where w_{val} is the number of instances where the value of var is equal to the value of val and $smoo$ is a small smoothing variable.

3.4.2 Credal sum-product network

CSPNs differ from SPNs in that they use credal sets for their edge weights, instead of single numerical values. As such, a lower and upper probability

can be taken for each edge. These can in turn be used to output a lower and upper probability in the root. These probabilities can then be used to do credal classification.

ϵ-contamination

One way of creating edge weights is by adding a generic ϵ-contamination, as done by Mauá et al. [22]. A normal SPN is learned from the LearnSPN algorithm, after which ϵ-contamination is used to generate a CSPN. However, this method relies on arbitrarily choosing the value for ϵ. As such, another method is chosen, which adds uncertainty to the model based on the missing data.

LearnCSPN

Levray and Belle created a credal variant on LearnSPN, LearnCSPN [17]. The main addition for LearnCSPN is that it is built to be more robust to missing data. When no data is missing the algorithm functions the same as LearnSPN. The paper also makes use of G-tests and EM, like Gens and Domingos [13].

G-test The G-test makes a slight change to Equation 3.11. Instead of the freq operator, it makes use of the mean of the lower count, $\underline{\text{freq}}$, and upper count, $\overline{\text{freq}}$, of a_1 and/or a_2. Resulting in the following equations when calculating the frequencies of (a_1, a_2):

$$\underline{\text{freq}}(a_1, a_2) = \text{freq}(a_1, a_2) \tag{3.13}$$

and

$$\overline{\text{freq}}(a_1, a_2) = \text{freq}(a_1, a_2) + \text{freq}(a_1, ?) + \text{freq}(?, a_2) + \text{freq}(?, ?) \tag{3.14}$$

where ? are missing values. The mean of these two is taken, after which the G-value is obtained. From this, the same evaluation is done as the G-test in LearnSPN.

EM-algorithm Due to the way subsets of T are made, some instances will be assigned to multiple clusters, instead of just one, based on their similarity. As such, subsets T_i can overlap. In the case of just two clusters, which is the only case used in this report for simplicity, this entails that all instances with missing data are grouped into both clusters. This results in the edge weight becoming an interval, namely $[\frac{|T_i|}{|T|}, \frac{|T_i|+|T_m|}{|T|}]$ where $|T_m|$ is the number

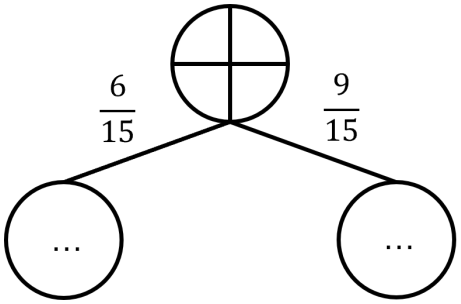

Figure 3.6: Visualisation for LearnSPN example

of instances that contain missing data. This overlap is the instances that contain missing data. This is the case as the models are set to only cluster into two clusters. With just two clusters, missing data can be imputed in a way that the instance can be part of either cluster, hence these instances are classified in both clusters.

3.4.3 Example

Just like the other examples, LearnSPN and LearnCSPN will be used on the example data. The algorithms will just be run over the attributes and not also the classes. In the implemented algorithms, each data set is first split over the classes to create a sum-node. This is done to make it easier to make inferences using the structure and was already implemented in DeeProb-kit [18].

First LearnSPN will be further explained. As there is more than 1 attribute, independence will have to be tested through the G-test:

$$G(A_1, A_2) = 2 \cdot (3.065 - 1.763 + 3.065) = 8.733$$

As the G-value is greater than 5 it is assumed that the attributes are dependent. Therefore the next step is to cluster the instances. By maximizing the log-likelihood 2 Gaussian distributions are created by the GMM, resulting in the instances being clustered into two clusters, one for all $(a_1, a_2) = (0, 0)$ and the rest in the other cluster. A sum-node is created with edges for each cluster with weights $\frac{6}{15}$ for the cluster containing instances with $(a_1, a_2) = (0, 0)$ and $\frac{9}{15}$ for the other cluster. This sum-node is visualized in Figure 3.6. The separate data sets can then be passed through the algorithm again until leaves can be generated.

As LearnCSPN works differently on data containing missing values, two instances will be added to illustrate how the different formulas work. These

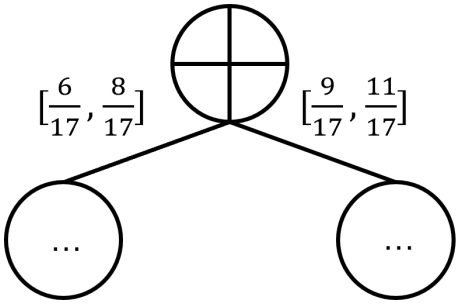

Figure 3.7: Visualisation for LearnCSPN example

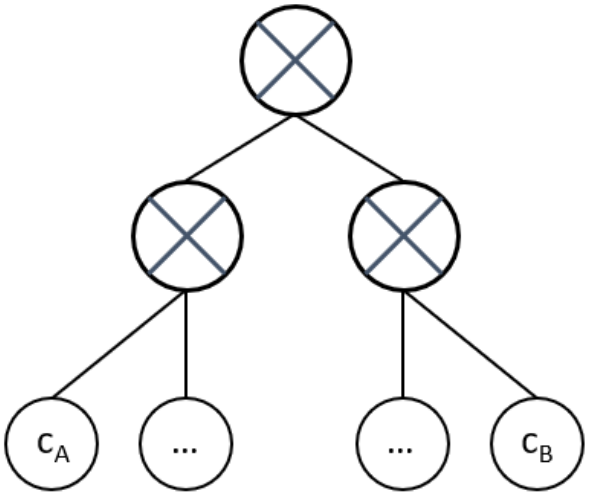

Figure 3.8: Example of the first two splits of LearnSPN by DeeProb-kit [18]

instances are $(a_1, a_2) = (?, 1)$ and $(a_1, a_2) = (?, ?)$. With these extra instances, the other G-test is run:

$$G(A_1, A_2) = 2 \cdot (3.449 - 1.738 - 0.839 + 3.373) = 8.491$$

Just as with LearnSPN the G-value is greater than 5 so some clustering will have to be done. The clustering results in the same clusters as for LearnSPN, with the addition that the two added instances with missing values are part of both clusters. This results in both clusters having these two instances and the edge weights becoming intervals, namely $[\frac{6}{17}, \frac{8}{17}]$ and $[\frac{9}{17}, \frac{11}{17}]$. This sum-node is visualized in Figure 3.7. The data set of each cluster will go through the process again until leaves can be formed.

DeeProb-kit [18] has some initial steps for creating an SPN to make inference easier, before generating based on LearnSPN. Figure 3.8 shows the top

part of such an SPN. The root node is a product-node where the data is split into subsets per class. Because of this split the class attribute of these data sets is always the same and is split from the rest of the data set to create a product-node with the specific class as a leaf node. The rest of the data set is then put through the algorithm to create a sub-(C)SPN, which continues from the nodes with dots in Figure 3.8.

The complete structure has leaves with likelihood functions. When an instance has to be predicted, it is put through the network bottom-up for each of the classes. This means that the likelihood for each class is calculated based on the instance to predict. These (interval) likelihoods can be evaluated as in Section 3.2.3.

Chapter 4

Methodology

To answer the research question and sub-questions, different credal classifiers will be compared on the metric of robustness. Robustness will be measured by looking at the differing accuracies and accuracy intervals over different levels of missing data.

4.1 Classifiers

The classifiers that will be evaluated are the naive Bayesian classifier (NBC), the naive credal classifier (NCC), decision tree classifiers built using the C4.5 and credal-C4.5 algorithms, the sum-product network (SPN), and the credal sum-product network (CSPN). These are three different classifiers with a classical and credal variant, making it possible to compare between classical and credal (SQ1), and between each different type of classifier (SQ3).

4.1.1 Implementations

Working implementations of the naive classifiers and C4.5 algorithms from papers were lacking, They were either written in software that is no longer supported, or in software that is not freely available. As such, the implementations of these models are made from scratch, based on the papers where they were first introduced.

For the creation of SPN the implementation of LearnSPN in *DeeProb-kit* [18] was used. More specifically, the implemented *SPNClassifier* was used to make classifications. The used model was run with maximum likelihood estimation for leaf creation, Gaussian mixture model with K-Means initialization for sum-node creation, and greedy variable splitting with a g-test for independence checking for product-node creation. This library was also

used as a basis for creating a CSPN classifier. The code was modified to use LearnCSPN based on the paper by Levray and Belle [17].

4.2 Experiments

All of the classifiers will be evaluated on data sets from the UCI repository [10]. Data sets from this repository are chosen, as they are used in a multitude of papers on credal classification [7, 19, 23]. The classifiers will be evaluated on two types of data sets. The first evaluation will take data sets without missing data. These data sets will then be modified with increasing levels of missing data to simulate how the classifiers respond to this missing data. In a second evaluation, data sets with missing data will be used, without generating extra missing data, to simulate a more realistic scenario. In the first evaluation, the data sets will have instances of which either none, 5%, 10%, 20%, or 30% contain missing data. This is done to evaluate SQ2. This data will be generated *completely at random* These percentages are based on the paper by Mantas et al. [20], where these were the percentages of noise that were added to the data. This missing data is only added to the data that is used for training the classifier. The classification is done on complete test data.

4.2.1 Handling missing data

The C4.5 algorithms can make decision trees even when there exists missing data in a data set. The same goes for LearnCSPN, as it has been specifically developed to generate CSPNs on data sets with missing data. However, the naive classifiers and classical LearnSPN cannot handle missing data when training the classifier by themselves. As such, mean imputation was used to impute the missing data for these models. Mean imputation was chosen as it imputes based on a simple characteristic of a column, instead of doing excessive modeling of instances compared to the rest of the data. Also, mean imputation is often used as a simple, yet good, imputation method.

4.2.2 LearnCSPN discrepancy

The network and hence the accuracies of the SPNs and CSPNs should be the same when there is no missing data in a data set when learned from LearnSPN and LearnCSPN. However, for the nursery dataset and the breast cancer data, this does not hold. After looking into the data sets and the code no bugs or oddities could be found to explain this difference. It is assumed

Predicted	True
Disease A	Disease A
Disease A	Disease B
Disease C	Disease C

Table 4.1: Example of predicted classes and true classes when sets cannot be predicted

that due to the bigger size of these data sets, some randomness is added in parts of the optimizations in the code that is not handled in the same way.

4.3 Generating Accuracies

Using the classifiers and data sets, it is possible to start generating results. Each classifier will be trained on the same data sets with the same missing data. The data sets will be split into 70% training data and 30% test data. These percentages are chosen to see how well the classifiers can handle new, unseen data. The training data is used to train the classifiers, whilst the test data is used to make predictions by feeding each instance to a classifier to retrieve a class, which can in turn be compared to the actual class to obtain the accuracy of the classifier.

4.3.1 Classical accuracy

In the case of the classical classifiers and the tree classifier from credal-C4.5, it is possible to get an accuracy value. This is obtained by dividing the total number of correct predictions by the total number of predictions. To give a concrete example, take the data in Table 4.1. Diseases A and C are correctly predicted, whilst disease B is not. As such the accuracy of the classifier is $\frac{2}{3}$.

4.3.2 Credal accuracies

For the NCC and the CSPN it is a bit more complicated. These credal classifiers can make set predictions, meaning that multiple classes can be predicted instead of just one. As such, the accuracies of these predictions will be shown with a lower and upper bound on the accuracy. The lower accuracy will be the accuracy of the predictions being exactly correct. This entails that if a set is predicted this is always considered as a false prediction. The upper accuracy is calculated by considering the set predictions which

Predicted	True
Disease A	Disease A
Disease B	Disease A
{Disease A, Disease B}	Disease C
{Disease B, Disease C}	Disease C
{Disease A, Disease B, Disease C}	Disease B

Table 4.2: Example of predicted classes and true classes when sets can be predicted

contain the correct class as correct predictions. With this lower and upper accuracy, it is possible to show an accuracy interval for the NCC and CSPN.

Next to the interval accuracies, the robust accuracy of the NCC and CSPN. This is the accuracy over just the instances that were not predicted as a set. These are the instances for which NCC and CSPN are fully certain. These instances are supposedly robust, as a single class can still be predicted, even when uncertainty was added through the imprecise Dirichlet model or missing data. Hence the name robust accuracy.

To make these different types of accuracy clearer, take the data in Table 4.2. The lower accuracy of the classifier can be calculated by finding the number of classes that are correctly classified, meaning that all set predictions are per definition wrong, as a set is not equal to a single value. In this case, this would result in only the first prediction being correct, meaning a lower accuracy of $\frac{1}{5}$. The upper accuracy will be calculated by also taking the cases where the true class is contained within a predicted set. As such, the last two predictions in Table 4.2 will also be seen as correct, resulting in an upper accuracy of $\frac{3}{5}$. This results in an accuracy interval of $[\frac{1}{5}, \frac{3}{5}]$ for this classifier. The robust accuracy is only measured over instances where no set predictions are made. As such, only the first two predictions are taken into account, of which only one is correct, resulting in a robust accuracy of $\frac{1}{2}$.

4.4 Discretization of data sets

As expected, the results show that the naive classifiers perform badly on data sets with many continuous attributes, as these attributes contain lots of different values. This results in the test set containing attribute values that were not in the train set. As such, the classifier has never seen some values, resulting in an inability to make predictions. On these data sets the accuracy of the NBC is very low and the intervals of the NCC are very wide, sometimes being the whole range of possible accuracies. Next to this, the

NCC is sometimes unable to do any robust classifications, meaning it cannot, with enough certainty, say something about the data that it is handed for classification. As such, the data sets on which the naive classifiers perform poorly due to the continuous attributes will be slightly modified to show some extra results. The continuous attributes will be discretized, after which both the original and the discretized data set will be run through the experiment.

4.5 Code availability

All code used can be found on GitHub [16]. A further explanation of the code and how to use the code for reproducing the results can be found in the README file of the repository, as well as on the main page of the repository. All raw results can be found in the "Results" folder.

Chapter 5

Results and Analysis

This section will present the results that were generated by running the experiments as mentioned in Section 4. Some follow-up analyses were done to further explore some results and to get a better understanding of what could be the underlying reasons for some of the results. All results will be presented in context to one of the sub-questions mentioned in Section 2.4.

5.1 Robustness of classifiers

By comparing the naive Bayesian classifier (NBC) and naive credal classifier (NCC) accuracies, it can be seen from Figure 5.1 that the classical accuracy falls within the credal interval accuracy. This is due to the use of the imprecise Dirichlet model (IDM) and the usage of just interval dominance and not credal dominance, as only the lower and upper probability are taken for evaluation. However, when comparing the classical accuracy and the credal robust accuracy, the NCC seems to outperform, or at least match the performance of the NBC in most cases. These conclusions are visualized in Figure 5.1. This trend showcases the robustness through conservativeness which was mentioned in Section 2.3.2.

This pattern of the classical accuracy falling within the lower and upper accuracy of the credal classifier is not seen for the sum-product networks (SPNs). This is not the case as SPNs and credal SPNs (CSPNs) are independently created structures. This results in different structures, which make different predictions, if the data set on which is trained contains missing data. For almost all data sets the CSPN accuracy interval contains or dominates the SPN accuracy, and the robust accuracy is greater than the classical accuracy, see Figure 5.2. This suggests that the CSPN makes better predictions when it is certain, and can make uncertainty clear in a well-performing in-

Data set name	$\|T\|$	$\|A\|$	$\|A_{cont}\|$	$\|A_{disc}\|$	$\|C\|$
Balance Scale	625	4	0	4	3
Car	1728	6	0	6	4
Cmc	1473	9	2	7	3
Dermatology	366	34	1	33	6
German credit*	1000	20	7	13	2
Glass*	214	9	9	0	7
Haberman	306	3	3	0	2
Hepatitis*	155	19	4	15	2
Horse colic*	368	26	7	19	2
House votes	435	16	0	16	2
Iris	150	4	4	0	3
Lymphography	146	18	0	18	4
Nursery	12960	8	0	8	4
Soybean	683	35	0	35	19
Wine*	178	13	13	0	3
Wisconsin breast cancer*	699	9	9	0	2
Zoo	101	16	0	16	7

*Data sets that have been discretized for the naive classifiers

Table 5.1: Characteristics of each data set

terval. This is the case on all levels of missing data. This suggests that the CSPN is more robust than the SPN.

However, the naive and tree classifiers handle missing data in the same way in both the classical and the credal variant. LearnSPN cannot handle missing data whereas LearnCSPN can. This results in an evaluation between one structure that is built on data where data was imputed and another that has added uncertainty based on the actual missing data. This results in dissimilar models being evaluated on a metric for which one is made and the other is not. This caveat should be taken into account when drawing conclusions regarding the robustness of SPNs and CSPNs.

As for the tree classifiers, conclusions on robustness are a bit harder to draw. The trees make singular classifications. This results in just being able to compare the accuracies of both trees. The trees from credal-C4.5 seem to match and in a few cases outperform the trees from C4.5, see Figure 5.3. Actual robustness is harder to measure. You could say that credal-C4.5 takes more uncertainty into account and thus is more certain about the made predictions. However, quantifying whether credal-C4.5 is more robust, and how much more robust it is, is hard. This is because the accuracy does not say something about properly handling uncertainty. It just shows the

performance of the classifier in the sense of correct predictions.

Given these results, take SQ1: *Are credal classifiers more robust than classical classifiers?*. Yes, credal classifiers are more robust, although quantifying this is hard. This is due to robust accuracy and classical accuracy being calculated on a different number of instances, and credal interval accuracies are not the same as classical unique accuracies. It is clear that credal classifiers take more uncertainty into account and act on this uncertainty. Usually, they perform similarly, if not better, and are more robust than their classical counterpart.

5.2 Classifiers on differing levels of missingness

When looking at the naive classifiers, as mentioned, the NBC accuracies fall within the accuracy intervals of the NCC. Thus drawing conclusions based on their accuracy for robustness is hard. Instead of looking at accuracies, take robust classifications. From Table 5.2 it can be concluded that there seems to be no clear relation between the NCC making robust classifications and the level of missing data. This is due to there not being a dependence between the interval created by IDM and the percentage of missing data.

Now take the average percentage of robust classifications by the CSPN over differing levels of missing data in Table 5.3. This shows a clear trend in most data sets of a decrease in the number of instances that are uniquely classified, i.e., classified robustly. This suggests that the more data is missing, the less certain the CSPN becomes, as it is more prone to make set classifications over unique classifications. This can be seen as the CSPN being robust to increasing levels of missing data. It was to be expected that CSPNs would handle uncertainty in some way, as LearnCSPN takes missing data into account when creating a CSPN.

The downward trend that is seen in Table 5.3, although clear, is not consistent over all data sets (e.g., German credit). Thus, even though there is a relation between missing data and robust classifications, other characteristics of the data sets also have some influence on robust classifications.

On the data sets with inherent missingness, the classifiers perform similarly to the way they perform on generated missing data, see Table 5.4. It should be mentioned that all classifiers perform poorly on the horse colic data set. This poor performance was not seen in other papers which made use of this data set. This decrease in performance can be seen for other data sets as well but is most extreme for the horse colic data set. The reason,

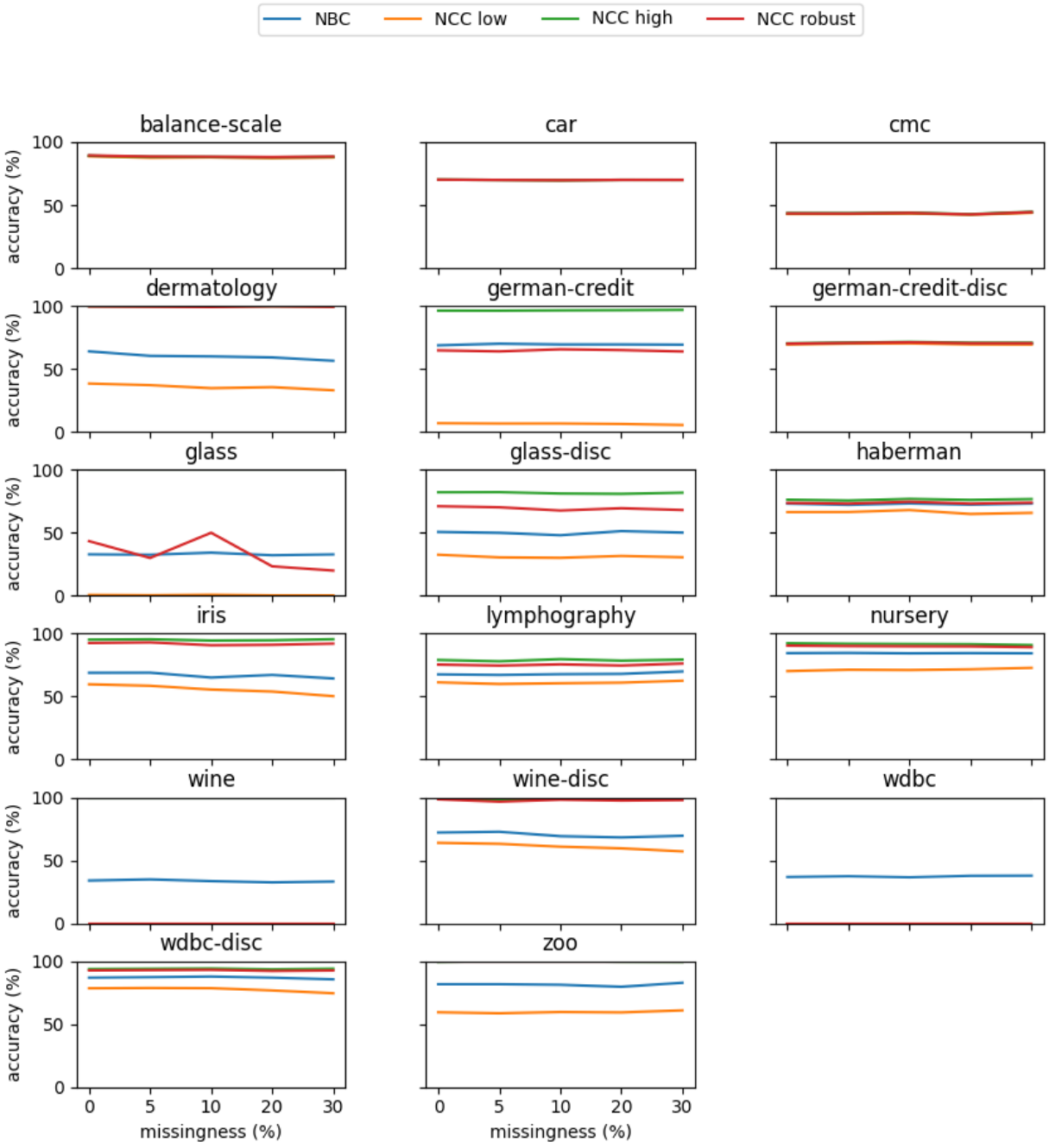

Figure 5.1: Plots of accuracies of naive classifiers per data set

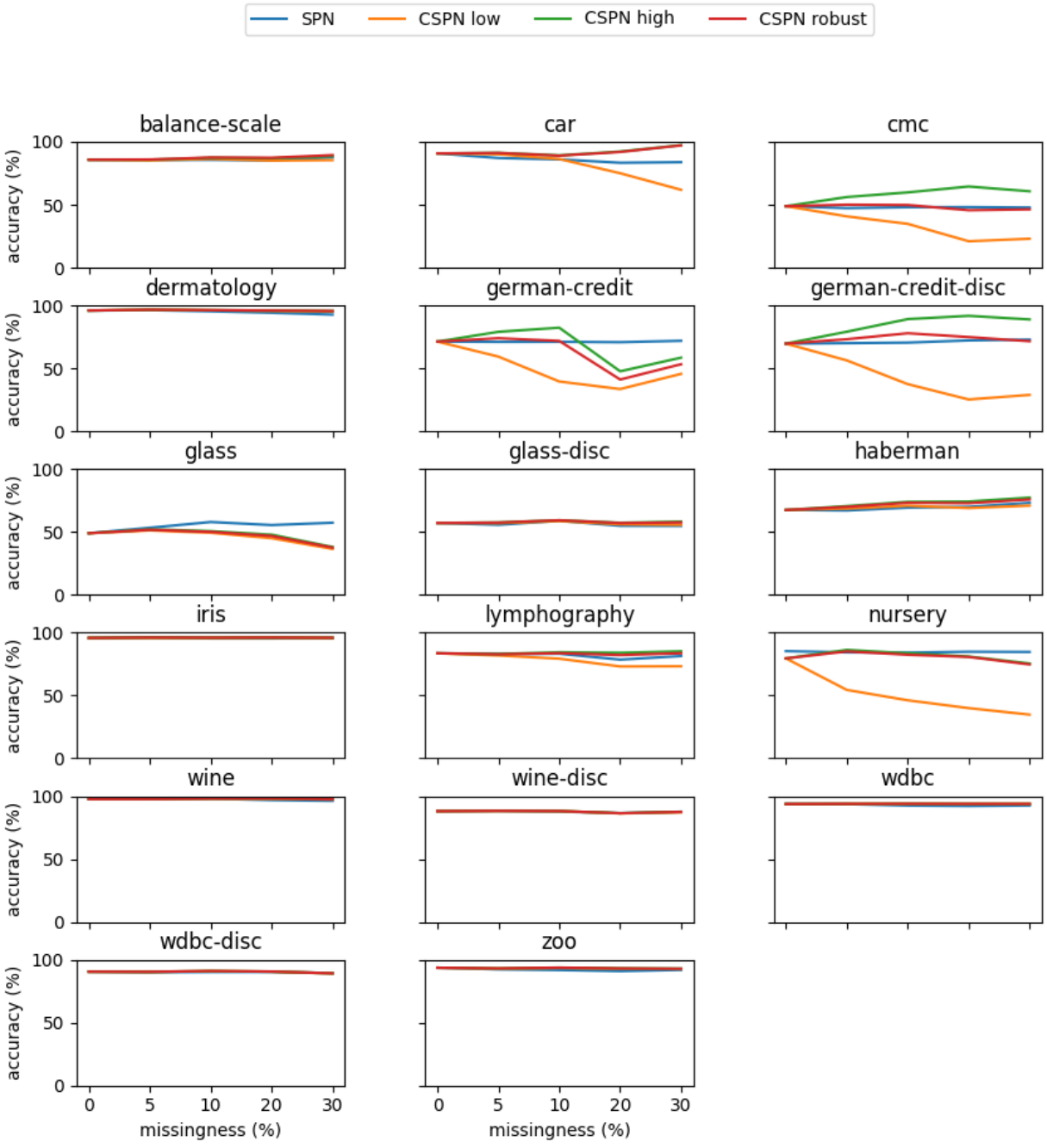

Figure 5.2: Plots of accuracies of SPN classifiers per data set

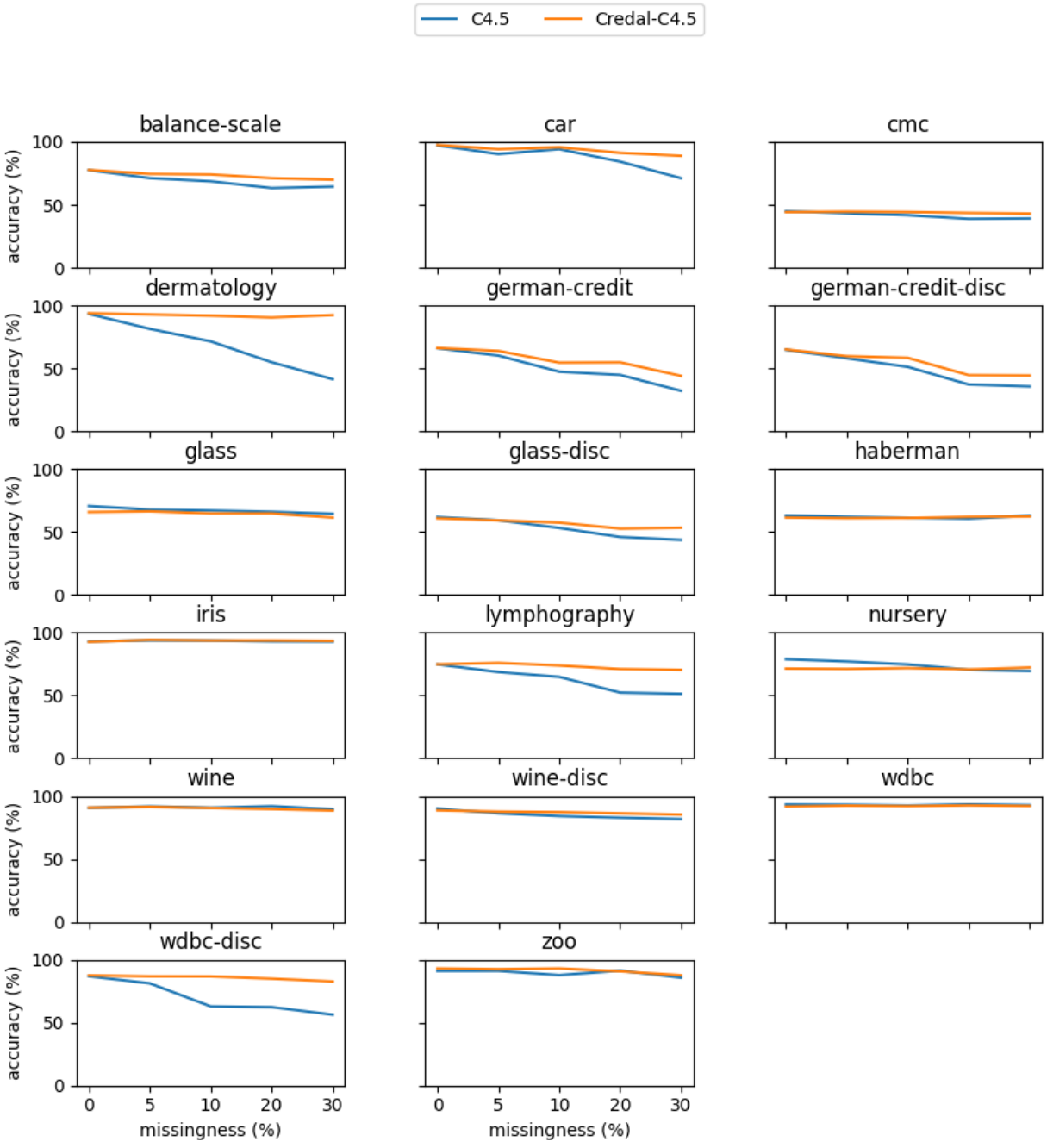

Figure 5.3: Plots of accuracies of tree classifiers per data set

Data set name	$\|T\|$	**0%**	**5%**	**10%**	**20%**	**30%**
Balance Scale	625	99,20	98,83	99,38	99,03	99,12
Car	1728	100,00	100,00	100,00	100,00	100,00
Cmc	1473	98,91	99,18	99,00	99,31	98,79
Dermatology	366	38,56	37,34	35,01	35,62	33,29
German credit	1000	10,71	10,44	10,22	9,70	8,66
German credit*	1000	99,10	99,00	99,20	98,90	98,60
Glass	214	1,54	1,54	1,54	1,54	1,54
Glass*	214	44,86	42,52	48,13	46,26	42,99
Haberman	306	90,14	90,95	91,12	88,83	89,05
Iris	150	64,57	63,03	61,22	59,29	54,66
Lymphography	146	81,28	80,47	80,19	81,80	82,16
Nursery	12960	77,61	79,16	79,07	79,81	81,57
Wine	178	0,00	0,00	0,00	0,00	0,00
Wine*	178	65,17	65,17	62,92	58,99	57,30
Wisconsin breast cancer	699	0,00	0,00	0,00	0,00	0,00
Wisconsin breast cancer*	699	83,98	84,12	83,55	83,26	82,40
Zoo	101	59,46	58,71	59,68	59,35	60,97

Table 5.2: Average percentage of robust classifications over all cross-validations per data set per level of missingness by the NCC

Data set name	$\|T\|$	**0%**	**5%**	**10%**	**20%**	**30%**
Balance Scale	625	100,00	99,13	98,65	97,40	95,54
Car	1728	100,00	98,84	97,07	81,62	63,70
Cmc	1473	100,00	81,46	69,92	46,03	49,76
Dermatology	366	100,00	100,00	99,91	99,84	99,72
German credit	1000	100,00	80,26	55,10	81,63	85,63
German credit*	1000	100,00	76,97	48,09	33,76	40,42
Glass	214	100,00	99,07	98,49	96,25	97,21
Glass*	214	100,00	99,13	98,71	98,30	96,77
Haberman	306	100,00	97,99	96,87	94,73	93,61
Iris	150	100,00	99,70	99,78	99,92	99,78
Lymphography	146	100,00	98,87	94,78	88,93	87,65
Nursery	12960	100,00	64,02	56,06	49,57	46,60
Wine	178	100,00	99,94	100,00	99,69	99,75
Wine*	178	100,00	100,00	100,00	99,88	99,70
Wisconsin breast cancer	699	100,00	99,98	99,96	99,96	99,98
Wisconsin breast cancer*	699	100,00	99,96	99,90	99,96	99,96
Zoo	101	100,00	99,77	99,77	99,67	99,68

Table 5.3: Average percentage of robust classifications over all cross-validations per data set per level of missingness by the CSPN

Data set name	Naive classic acc	credal low	credal high	credal robust	C4.5 classic acc	credal acc	SPN classic acc	credal low	credal high	credal robust
Hepatitis	79,72	6,17	99,79	97,11	45,32	51,35	86,52	85,60	87,66	87,44
Hepatitis*	79,29	40,28	94,47	87,83	45,11	54,11	82,55	82,84	84,89	84,60
Horse colic	12,22	0,37	99,67	30,00	5,26	34,96	69,15	19,93	21,07	26,13
Horse colic*	19,15	1,11	99,67	70,00	3,48	36,30	62,37	40,70	43,26	42,84
House votes	89,90	89,85	89,92	89,92	94,33	93,82	94,99	94,99	94,99	94,99
Soybean	56,74	39,75	90,25	81,05	56,88	67,42	88,57	88,03	88,24	88,38

Table 5.4: Accuracies on data sets with inherent missing data

probably, is that this thesis makes a split of 70/30 instead of 90/10 for the train/test split of the data sets. Mantas et al. [20] use this 90/10 split to evaluate C4.5 and credal-C4.5 on some of the same data sets as used in this thesis. For almost all data sets the algorithms perform better with the 90/10 split than the 70/30 split, when looking at 0% noise and missing data.

To conclude on SQ2: *What is the difference in robustness between credal classifiers and their classical counterpart under varying levels of missing data?*, the NCC seems to be more robust than the NBC. However, there does not seem to be a clear relation between robustness and the level of missingness. The robustness comes from the added conservativeness of the IDM, not the missing data. The CSPN, however, is more robust than the SPN. The robustness has an explicit relation to the level of missingness. The CSPNs show that they take the uncertainty from missing data into account, and adjust their predictions accordingly. They make less unique classifications with added uncertainty, and could thus be seen as more robust.

5.3 Differences between similar classifiers

Both SPNs and CSPNs seem to outperform both the naive and the tree classifiers on all data sets on all levels of missing data, as can be seen in Figures 5.4 and 5.5. This is probably since SPNs are more complex classifiers than the trees built by the C4.5 algorithms or the simple naive classifiers. The naive classifier can only classify data that it has already seen, whereas SPNs are built on distributions of data. This makes SPNs able to make predictions on instances it has never seen, based on the distribution it created from the training data. Also, naive classifiers assume independence. Learn(C)SPN does independence tests to take possible dependencies into account. The

C4.5 algorithms only have a single metric, the split criterion, from which the tree is built. Learn(C)SPN uses clustering, independence tests, and maximum likelihood estimations to create a structure for classification. Therefore, it was expected that the more complex SPNs would outperform the other simpler classifiers.

Decision trees from the C4.5 algorithms and the naive classifiers have no clear relations with each other. On each separate data set the classical variant and credal variant perform similarly to each other. However, on some data sets the naive classifiers outperform the decision trees, and sometimes it is the other way around. There does not seem to be a visible relation between the type of data and whether either type of classifier performs better.

On the data sets that had to be discretized, the naive classifiers performed poorly. These data sets contained mostly continuous attributes. These attributes have unique values for almost each instance. As such, the trained naive classifier has to make predictions on instances with values it has never seen, and thus cannot predict. After discretization, the naive classifiers performed better, whilst the reverse is true for the tree classifiers and SPNs. The discretization results in underlying information being discarded by "simplifying" attribute values. As such, the tree classifiers and SPNs learn less about the underlying distribution of the data set, which makes it harder for them to make correct predictions.

To answer SQ3: *Are there similar differences in performance on differing levels of missing data between the models in their classical form, as well as their credal form?*, yes there are similar differences between different types of classifiers in their classical and credal form on differing levels of missing data. The different classifier types do not show the same differences between them over different data sets. However, the differences that are seen, are similar between the classical and credal variants of each classifier type when considering each data set separately.

5.4 More credal classifiers

Lastly SQ4: *Do there exist more classifiers, that have no credal variant yet, that can be made credal for more robust classifications?*. This question is more exploratory than the others. This was done to answer whether credal classification can be generalized.

One option of making classical classifier credal is through the use of the IDM. This was done to create the NCC, and many other credal classifiers sprang from it. This usage of the IDM does make a credal classifier, however, the robustness and performance of the credal classifier are heavily dependent

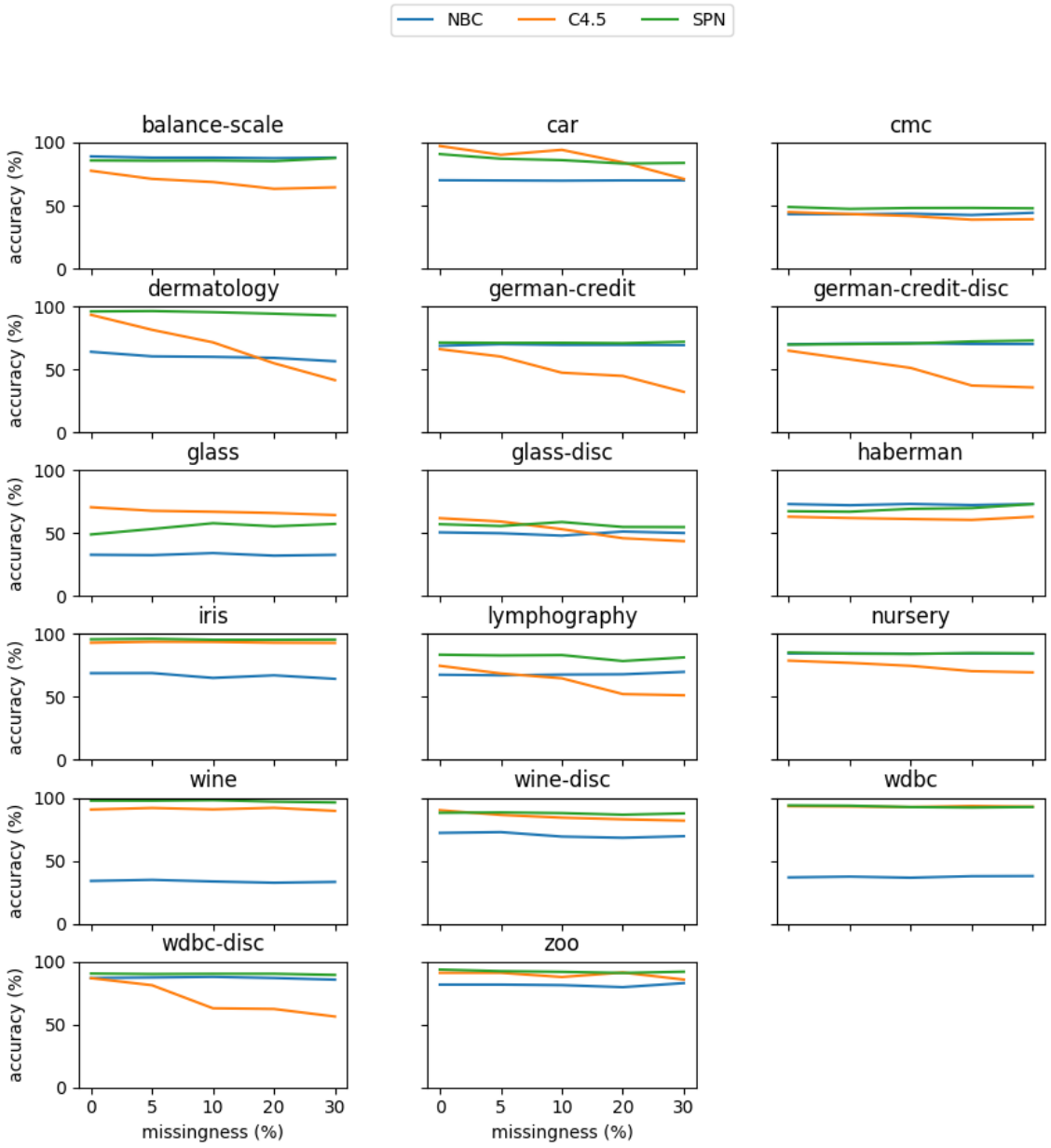

Figure 5.4: Plots of accuracies of classical classifiers per data set

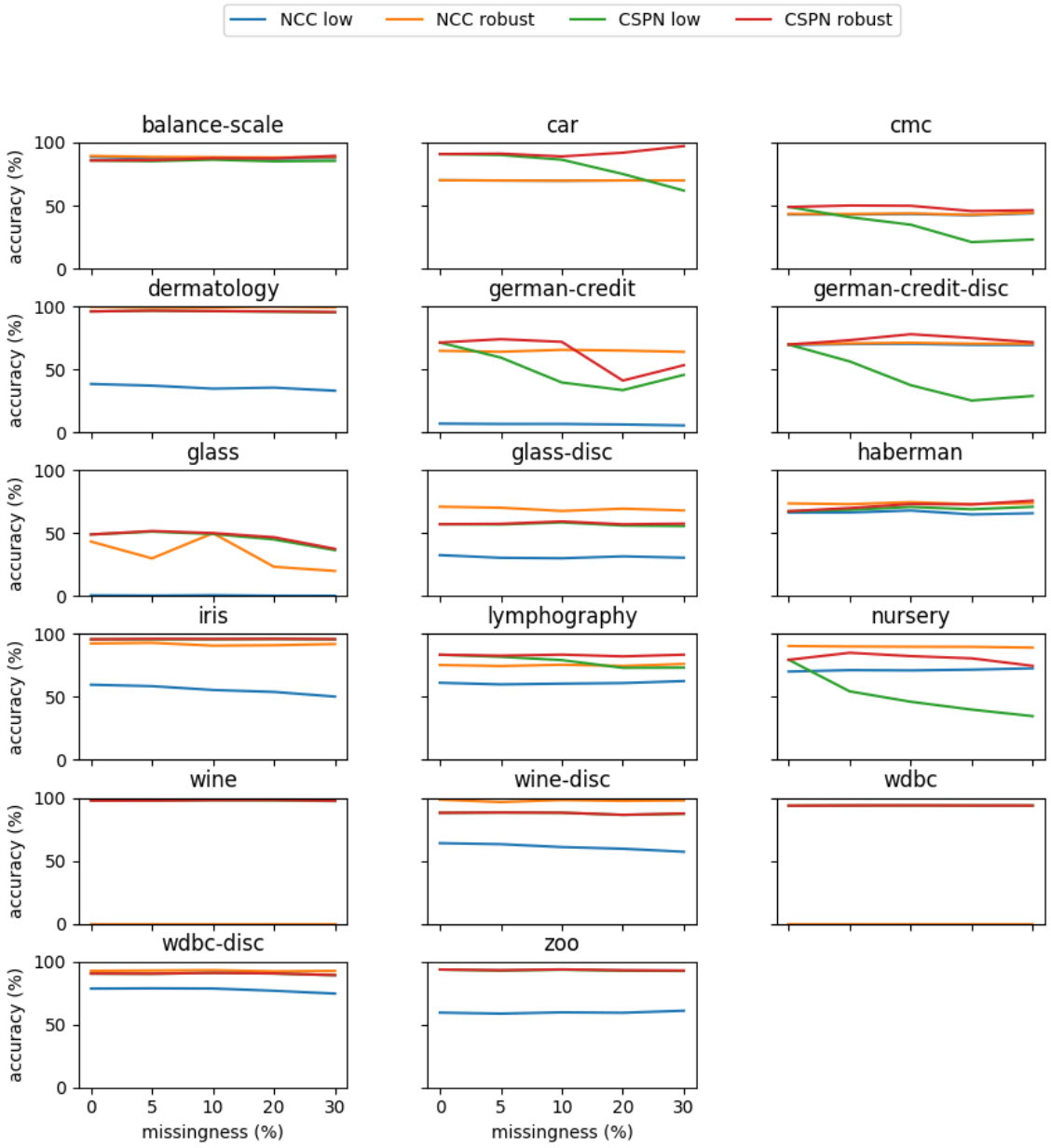

Figure 5.5: Plots of accuracies of credal models per data set

on the optimization of parameter s for the IDM. In the literature, it is suggested that s is optimal between 1 and 2 [39]. This seems a bit short-sighted. s determines the number of pseudo-instances that are used to add uncertainty. When a data set is large this number between 1 and 2 is quite small for some data sets, resulting in a classifier that is very close to a classical classifier. As such, if you were to create a credal classifier from a classical classifier using the IDM, then you should optimize s for each data set separately.

Mantas et al. [20] attempt to find an optimal value for s, regarding credal-C4.5, by varying the value between 0.25 and 4. They conclude that s has to be optimized for each data set, but do not give clear relations between the value s and some data characteristics.

When talking about credal classifiers being more robust variants than their classical counterparts, then the ones that come from probabilistic classifiers are best to talk about. Credal tree classifiers are potentially more robust. They make use of credal sets for the generation of the tree, but this tree has no uncertainty in the paths from the root to leaf or in the leaves. This results in fully certain predictions. In turn, this results in accuracy being the only measure to compare performance and robustness. Robustness is easier to argue when the classifier is allowed to make set classifications.

Given all this, the most proper way to create a credal classifier would be to take a probabilistic classifier that is incapable of handling uncertainty in data and make it able to handle this through the theory of credal sets. LearnCSPN does this by making a CSPN that was built based on data with missing data. However, another way is to take an SPN and contaminate it to find robust instances, as done by Mauá et al. [22]. This results in a structure that can classify multiple classes if needed, and has uncertainty, either from the data or generated randomly.

To make a concrete suggestion for extensions of credal classification, ensemble methods can be used to create credal sets, by making an ensemble of probability distributions from multiple probabilistic classifiers. This credal set can be used to make credal classifications. Some assumptions might have to be added, to not create an overly general classifier that effectively cannot uniquely classify any instance. Another simpler option is to take an ensemble method with a voting system and delete the voting system entirely. This will also lead to set classifications.

Another option would be to take a graphical probabilistic classifier, like an SPN, and make it handle missing data, as LearnCSPN did for SPNs. For instance, a probabilistic decision graph [15] is a type of graphical probabilistic classifier that does not seem to handle missing data yet and could benefit from the theory of credal sets to add to this. By adding uncertainty to the

representation of the probability distribution, a credal set can be generated for making predictions through credal classification.

Chapter 6

Conclusion

This thesis tries to answer the question: *When and how, in relation to missing data, are credal classifiers more robust than their non-credal counterparts, and to what degree does this generalize?* To do this, the main research question was subdivided into four sub-questions, which were answered in Section 5. This section will summarize the results and answers to the sub-questions to answer the main question of this thesis.

First the when and how. Credal classifiers are almost always more robust than their classical counterparts. This is the case for most data sets and under each level of missing data, even when there is no missing data. This robustness mostly comes from the conservative nature of credal classifiers. They either generate uncertainty or take the actual uncertainty in data into account to create credal sets, so credal classification can be made possible. The property of being able to make set classifications makes it easier to be more robust than classical classifiers. This is the case as classical classifiers can only uniquely predict classes, resulting in either correct or incorrect classifications. Credal classifiers have the benefit of being partly correct when classifying the correct class in a set with other classes.

But does this generalize? In this thesis, multiple classifiers were compared and for these classifiers, yes the robustness generalizes. The credal classifiers can make set classifications, resulting in a better representation of the uncertainty, coming from missing data, in the prediction of the credal classifier. However, whilst the three classifiers discussed in this paper have all benefited from the theory of credal sets, more classifiers could benefit from this robustness. It seems that the theory could be applied well to classical probabilistic classifiers. These could be taken as a starting point to further generalize credal classifiers and credal classification for robustness.

This thesis attempts to better define robustness in the domain of (credal) classification. It has shown that credal classifiers can indeed be more robust

when trained on data sets containing either generated or inherent missing data. These findings generalize well over the three different classifiers in this report, and might even generalize to more (credal) classifiers.

Chapter 7

Discussion

This section is divided into two parts. First, the limitations of this thesis. In the process, some decisions had to be made to make the work viable within the given time limit. And second, suggestions of future work that could be done by future researchers, based on this work and things that were learned from studying the literature.

7.1 Limitations

There exist more classical classifiers with credal variants than are evaluated in this report, some of which are mentioned in Section 2. The ones that were chosen were taken to showcase classifiers of differing levels of complexity. However, given the time constraint of this thesis, only one truly complex model was used. The naive classifiers and tree classifiers are quite simple and basic classifiers that have seen improvements in recent years. More state-of-the-art models might have given a better view of where the domain of credal classifiers is at right now.

Using less complex models also resulted in the need to run experiments on less intricate data sets. Mainly the size of the data sets could not be too big, as the models were not optimized for running these. With more time and more optimizations, it would have been interesting to experiment with new, more complex data sets to see how well the drawn conclusions generalize to those kinds of data sets.

All models with hyperparameters used the same values for each data set. These hyperparameters are s for the imprecise Dirichlet model in the naive credal classifier and credal-C4.5, and the minimum numbers of columns and rows needed to split and some initialization values in LearnSPN and LearnC-SPN. Some testing was done to optimize hyperparameters when validating

the models, however finding the best hyperparameters for each model for each data set would take an extensive amount of extra time, which was not available. Arguably, finding optimal hyperparameters would also be subject to what one would like to optimize. In the case of credal classifiers, this optimal can take two forms. Either you want to maximize the accuracy on uniquely classified instances, or you would want to have better upper accuracy, suggesting that each instance is at least classified correctly in a set.

Early in the process of making this thesis, the decision was made to use tree classifiers from C4.5 and credal-C4.5. After already implementing and testing the algorithms, it was found that credal tree classifiers do not make credal classifications in the way that the models can be evaluated on robustness similarly. As time was already invested in implementing the model, it was decided to keep going with the tree classifiers. This led to results for the tree classifiers that did not have a clear distinction between classical and credal classifiers, as the credal tree cannot make set classifications. Because of this, evaluating the differences between probabilistic classifiers and tree classifiers was kept to a minimum, due to the difficulty of comparing different measures of accuracy.

7.2 Future work

This thesis mainly focused on creating classifiers from data that contains missing data, whilst the data that would be classified did not contain missing data. It would be interesting to see credal classifiers show more uncertainty when instances with missing data have to be classified. This could be done by training classifiers in the same way as this report does, or by training on data without missing data. A credal classifier that can be generated on missing data and can make classifications on instances with missing data while representing the uncertainty of both the training and classification seems like the most robust way to go about handling missing data.

In future work, more attention could be paid to state-of-the-art classifiers and data sets. The classifiers in this paper differ heavily in complexity, meaning some classifiers take more aspects of the data into account when training and making predictions. This makes it harder to compare the different types of classifiers. Similarly complex models could be evaluated to make a better comparison of their robustness and differences. To compare the classifiers, the UCI data sets can still be used for basic comparisons. However, it would be better to take more recent and sizable data sets to make a better case of the practical usability of credal classifiers.

Lastly, more of a recommendation, using similar classifiers would be best

when comparing them on robustness. Not each classifier that is called a credal classifier makes credal classifications as defined in this report. Credal classifiers that come from probabilistic classifiers or tree classifiers still make predictions in different ways, which should be considered when selecting classifiers to compare.

Bibliography

[1] Joaquín Abellán and Andrés R. Masegosa. An ensemble method using credal decision trees. *European Journal of Operational Research*, 205(1): 218–226, 8 2010. doi:10.1016/J.EJOR.2009.12.003.

[2] David Alvarez-Melis and Tommi S. Jaakkola. On the Robustness of Interpretability Methods. *CoRR*, 6 2018. doi:10.48550/arxiv.1806.08049.

[3] Alessandro Antonucci, Cassio P. de Campos, and Marco Zaffalon. Probabilistic graphical models. *Introduction to Imprecise Probabilities*, pages 207–229, 8 2014. doi:10.1002/9781118763117.CH9.

[4] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *Proceedings - IEEE Symposium on Security and Privacy*, pages 39–57, 6 2017. doi:10.1109/SP.2017.49.

[5] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On Evaluating Adversarial Robustness. *CoRR*, 2 2019. doi:10.48550/arxiv.1902.06705.

[6] G. Corani and A. Antonucci. Credal ensembles of classifiers. *Computational Statistics & Data Analysis*, 71:818–831, 3 2014. ISSN 0167-9473. doi:10.1016/J.CSDA.2012.11.010.

[7] Giorgio Corani and Marco Zaffalon. Lazy naive credal classifier. *Proceedings of the 1st ACM SIGKDD Workshop on Knowledge Discovery from Uncertain Data, U'09 in Conjunction with KDD'09*, pages 30–37, 2009. doi:10.1145/1610555.1610560.

[8] Fabio G. Cozman. Credal networks. *Artificial Intelligence*, 120(2):199–233, 7 2000. doi:10.1016/S0004-3702(00)00029-1.

[9] Geoff Delaney, Michael Barton, Susannah Jacob, and Bin Jalaludin. A model for decision making for the use of radiotherapy in lung can-

cer. *The Lancet Oncology*, 4(2):120–128, 2 2003. doi:10.1016/S1470-2045(03)00984-7.

[10] Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[11] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29(2-3):131–163, 1997. doi:10.1023/A:1007465528199.

[12] Ashutosh Garg and Dan Roth. Understanding Probabilistic Classifiers. In Luc De Raedt and Peter Flach, editors, *Machine Learning: ECML 2001*, pages 179–191, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. doi:10.1007/3-540-44795-4_16.

[13] Robert Gens and Pedro Domingos. Learning the Structure of Sum-Product Networks, 2013. URL `https://proceedings.mlr.press/v28/gens13.html`.

[14] Ronan Hamon, Henrik Junklewitz, and Ignacio Sanchez. Robustness and explainability of artificial intelligence. *Publications Office of the European Union*, 2020. doi:10.2760/57493.

[15] Manfred Jaeger. Probabilistic decision graphs - Combining verification and AI techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, 12 (SUPPL. 1):19–42, 1 2004. doi:10.1142/S0218488504002564. URL `www.worldscientific.com`.

[16] Rob Janssen. GitHub Repository, 2023. URL `https://github.com/Nihrin/Master_Thesis`.

[17] Amelie Levray and Vaishak Belle. Learning Credal Sum-Product Networks. In *Automated Knowledge Base Construction (2020)*, 1 2019. doi:10.48550/arxiv.1901.05847.

[18] Lorenzo Loconte and Gennaro Gala. DeeProb-kit: a Python Library for Deep Probabilistic Modelling, 12 2022.

[19] Carlos J. Mantas and Joaquín Abellán. Credal-C4.5: Decision tree based on imprecise probabilities to classify noisy data. *Expert Systems with Applications*, 41(10):4625–4637, 8 2014. doi:10.1016/J.ESWA.2014.01.017.

[20] Carlos J. Mantas, Joaquín Abellán, and Javier G. Castellano. Analysis of Credal-C4.5 for classification in noisy domains. *Expert Systems with Applications*, 61:314–326, 11 2016. doi:10.1016/J.ESWA.2016.05.035.

[21] Denis Deratani Mauá and Fabio Gagliardi Cozman. Thirty years of credal networks: Specification, algorithms and complexity. *International Journal of Approximate Reasoning*, 126:133–157, 11 2020. doi:10.1016/J.IJAR.2020.08.009.

[22] Denis Deratani Mauá, Diarmaid Conaty, Fabio Gagliardi Cozman, Katja Poppenhaeger, and Cassio Polpo de Campos. Robustifying sum-product networks. *International Journal of Approximate Reasoning*, 101:163–180, 10 2018. doi:10.1016/J.IJAR.2018.07.003.

[23] S. Moral-García, Carlos J. Mantas, Javier G. Castellano, María D. Benítez, and Joaquín Abellán. Bagging of credal decision trees for imprecise classification. *Expert Systems with Applications*, 141:112944, 3 2020. doi:10.1016/J.ESWA.2019.112944.

[24] Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. *Proceedings of the IEEE International Conference on Computer Vision*, pages 689–690, 2011. doi:10.1109/ICCVW.2011.6130310.

[25] Erik Quaeghebeur and Vu-Linh Nguyen. Lecture Credal Sets, 2AMU30: Uncertainty Representation and Reasoning, 2022.

[26] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993. doi:10.1007/BF00993309.

[27] J R Quinlan. Improved Use of Continuous Attributes in C4.5. *Journal of Artiicial Intelligence Research*, 4:77–90, 1996.

[28] I Rish. An empirical study of the naive Bayes classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 3(22):41–66, 2001.

[29] Salvatore Ruggieri. Efficient C4.5. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):438–444, 3 2002. doi:10.1109/69.991727.

[30] S. Rasoul Safavian and David Landgrebe. A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):660–674, 1991. doi:10.1109/21.97458.

[31] Raquel Sanchez-Cauce, Iago Paris, and Francisco Javier Diez. Sum-Product Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3821–3839, 7 2022. doi:10.1109/TPAMI.2021.3061898.

[32] Katarzyna Stapor. Evaluating and comparing classifiers: Review, some recommendations and limitations. *Advances in Intelligent Systems and Computing*, 578:12–21, 2018. doi:10.1007/978-3-319-59162-9_2/TABLES/1.

[33] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is Robustness the Cost of Accuracy? – A Comprehensive Study on the Robustness of 18 Deep Image Classification Models. *CoRR*, abs/1808.0:631–648, 2018. doi:10.48550/arXiv.1808.01688.

[34] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini Google, Brain Benjamin Recht, and Ludwig Schmidt. Measuring Robustness to Natural Distribution Shifts in Image Classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020. doi:10.48550/arXiv.2007.00644.

[35] The Merriam-Webster.com Dictionary. Robustness, 2023. URL `https://www.merriam-webster.com/dictionary/robustness`.

[36] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. *7th International Conference on Learning Representations, ICLR 2019*, 5 2018. doi:10.48550/arxiv.1805.12152.

[37] Peter Walley. Inferences from Multinomial Data: Learning About a Bag of Marbles. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):3–34, 1 1996. doi:10.1111/J.2517-6161.1996.TB02065.X.

[38] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R. Salakhutdinov, and Kamalika Chaudhuri. A Closer Look at Accuracy vs. Robustness. *Advances in Neural Information Processing Systems*, 33:8588–8601, 2020. doi:10.48550/arXiv.2003.02460.

[39] Marco Zaffalon. The naive credal classifier. *Journal of Statistical Planning and Inference*, 105(1):5–21, 6 2002. doi:10.1016/S0378-3758(01)00201-4.

[40] Marco Zaffalon. Credible classification for environmental problems. *Environmental Modelling & Software*, 20(8):1003–1012, 8 2005. doi:10.1016/J.ENVSOFT.2004.10.006.

[41] Marco Zaffalon and Enrico Fagiuoli. Tree-Based Credal Networks for Classification. *Reliable Computing 2003 9:6*, 9(6):487–509, 12 2003. doi:10.1023/A:1025822321743.