Improving Artist Content Matching with Stacking

A comparison of meta-level learners for stacked generalization

FANNAR MAGNÚSSON
Improving Artist Content Matching with Stacking

A comparison of meta-level learners for stacked generalization

FANNAR MAGNÚSSON
Abstract

Using automatic methods to assign incoming tracks and albums from multiple sources to artists entities in a digital rights management company, where no universal artist identifier is available and artist names can be ambiguous, is a challenging problem. In this work we propose to use stacked generalization to combine the predictions of heterogeneous classifiers for an improved quality of artist content matching on two datasets from a digital rights management company. We compare the performance of using a non-linear meta-level learner to a linear meta-level learner for the stacked generalization on the two datasets, as well as on eight additional datasets to see how well our results generalize. We conduct experiments and evaluate how the different meta-level learners perform, using the base learners’ class probabilities or a combination of the base learners’ class probabilities and original input features as meta-features.

Our results indicate that stacking with a non-linear meta-level learner can improve predictions on the artist chooser problem. Furthermore, our results indicate that when using a linear meta-level learner for stacked generalization, using the base learners’ class probabilities as meta-features works best, while using a combination of the base learners’ class probabilities and the original input features as meta-features works best when using a non-linear meta-level learner. Among all the evaluated stacking approaches, stacking with a non-linear meta-level learner, using a combination of the base learners’ class probabilities and the original input features as meta-features, performs the best in our experiments over the ten evaluation datasets.

Keywords: machine learning; stacked generalization; combining classifiers; meta-learning; name disambiguation.
Referat


Nyckelord: maskininlärning; staplade generaliseringar; kombinerande klassificerare; metainlärning; namn disambiguering.
Acknowledgements

I would like to start by expressing my sincere gratitude to Jon Áslund, my industrial supervisor, and professor Henrik Boström, my academic supervisor. Their continuous guidance and support throughout the entire thesis work was invaluable. Without them, this thesis would not have been possible.

I would like to thank professor Sarunas Girdzijauskas for his guidance and for being my examiner. I would also like to thank my friends and classmates for all their support, and to thank my thesis opponent, Jonas Wallmeier, for helpful comments at the end of the thesis work.

Finally, I would like to extend my gratitude to my girlfriend, Helga Margrét Ólafsdóttir, for reading over the thesis a number of times, and for her invaluable support. Furthermore, I would like to extend my gratitude to my family for their endless support, and a special thank you to my mother, Jóhanna Rós F. Hjaltaín, who is always there for me.

Stockholm, June 27, 2018
Fannar Magnússon
# Contents

1 Introduction .......................... 1
  1.1 Background .......................... 2
  1.2 Problem .......................... 3
  1.3 Purpose .......................... 4
  1.4 Goal .............................. 4
     1.4.1 Benefits .......................... 5
     1.4.2 Ethics .......................... 5
     1.4.3 Sustainability .................... 6
  1.5 Methodology ....................... 6
     1.5.1 Philosophical assumptions ............ 7
     1.5.2 Research methods .................. 8
     1.5.3 Research approaches ................. 9
  1.6 Delimitations ..................... 9
  1.7 Outline (Disposition) ............. 10

2 Background ......................... 11
  2.1 Name disambiguation ............... 11
  2.2 Machine learning .................. 12
     2.2.1 Logistic regression ................. 12
     2.2.2 Naive Bayes ........................ 14
     2.2.3 Decision trees ...................... 15
     2.2.4 Artificial neural network .......... 16
  2.3 Ensemble methods .................. 18
     2.3.1 Bagging .......................... 19
     2.3.2 Boosting .......................... 19
     2.3.3 Stacked generalization .......... 20
  2.4 Related work ..................... 22

3 Methods ............................. 25
  3.1 Research strategies ............... 25
  3.2 Data collection .................... 26
  3.3 Data analysis ..................... 27
  3.4 Quality assurance .................. 28
Chapter 1

Introduction

The digitization of media industries has revolutionized how we interact with and consume media, like audio and video. In the last decade, on-demand streaming services such as Netflix and Spotify have made it easier than ever for people to access their favorite movies and music. Everything is now available on any device, wherever you are. We have seen major transformations of these industries, going from physical sales of movies and records, to digital downloads, to streaming services with millions of subscribers. Netflix, one of the leading video-on demand streaming services, has for example 125 million users [1], and Spotify, the leading music streaming service, has 170 million users [2]. Services such as these have large digital catalogues that are managed, maintained, and served to users located all over the world.

For a digital rights management company (DRMC) that manages a catalogue of millions of songs, there are many interesting challenges that follow. One of them being the task of mapping new incoming tracks/albums, from multiple sources, to correct artists in the DRMC’s system. The audio data being ingested into the system includes relevant metadata, such as information about the recording, the name of the artist and the name of the tracks/album. There is, however, no standard universal artist identifier available that can be included in the metadata, so when the DRMC is determining which artist entity in their system they should assign the incoming tracks/albums to, they have to rely on the artist name. The problem is however, that in the DRMC’s system, there might be multiple artists with the same name, so choosing the right artist can be difficult. This problem is generally known as name disambiguation, and can be solved either manually or automatically. In this case, with thousands of tracks/albums being ingested into the DRMC’s service each day, manually resolving these artist name disambiguation problems is inefficient. This problem can be automated using machine learning techniques.

The degree project was conducted with a DRMC. It builds on the previous work that has been done at the company on a solution called the artist chooser: a Random Forest machine learning model trained to choose the most likely artist, or create a new artist, to assign incoming tracks and albums to. With our work we aim to improve the current solution using advanced machine learning techniques. More
precisely, we propose to use stacked generalization [3] for improving the quality of artist content matching. We investigate and evaluate different stacking approaches and compare the performance of a non-linear meta-level learner to a linear meta-level learner for stacked generalization. To find out how well our solution on the artist chooser problem generalizes, we additionally evaluate our stacking approaches on eight datasets from the UCI Machine Learning Repository [4].

1.1 Background

The work in this thesis is concerned with using machine learning techniques for improving artist content matching. The field of machine learning gives computers the ability to learn from data, and understand data, using statistical techniques. The two main learning methods are supervised learning and unsupervised learning.

In supervised learning, a model is built to predict, or estimate, an output based on one or more inputs. In unsupervised learning, unlike supervised learning, no outputs are given with inputs at training time; still, the model can find structure in its inputs and learn relationships [5].

Name disambiguation problems, such as the artist chooser problem, are well known and are not restricted to a single context. These problems generally need to be solved for every service that manages a (digital) library, where the data can be ingested from multiple sources and there is no clear way to identify the incoming data and determine where to place it. The problem of choosing the right artist to place a track/album to, can be compared to the problem of assigning a work of literature to the correct author. A survey on methods for solving the author name disambiguation problem [6] covers automatic methods that can be used to solve it, namely supervised and unsupervised machine learning methods. Another paper on the issue by Smalheiser and Torvik [7] argues that each problem is different, but in general, supervised approaches perform better as they are tuned specifically to the data. Then again, a sufficient amount of labeled training data is needed.

In our case, the DRMC has a big catalogue of tracks and albums linked to artists in their system. This data can be used as training data and therefore a supervised approach has been chosen to solve the artist chooser problem. Currently, a Random Forest model [8] is trained on more than one million examples of albums and tracks. The training set includes a total of 47 features, including information such as the album’s licensor, label and feed, release date, language and country code of tracks, artist name length, and more. In general, the artist chooser problem is a binary classification problem, meaning that the training data is either labeled as do not assign (-1) or assign (1). For an album that belongs to an artist that has an ambiguous name, the correct artist is labeled as assign (1) and all other candidate artists with the same name are labeled as do not assign (-1) for that album.

The current artist chooser solution has worked okay, but its performance seems to be degrading. More and more tracks and albums are showing up on incorrect artist pages on the DRMC’s service and need to be manually corrected. Further-
1.2. PROBLEM

more, the performance of the current model is unknown, as it is not being validated in any way.

In this thesis project, we look into ways to improve the current artist chooser solution using machine learning techniques. We propose to use stacked generalization (stacking) [3] as a solution to improve the quality of the artist content matches. Stacking is one form of ensemble learning [9], a system where a combination of multiple learning algorithms is used to get a better predictive performance than for a single learner. Unlike other forms of ensemble based systems, such as bagging [10] and boosting [11], where the ensemble is constructed using homogeneous learning algorithms (same type of learning algorithms) and the training data is re-sampled; stacking is constructed using a number of heterogeneous learning algorithms [12] (different types of learning algorithms) that serve as base learners, and a meta learner that combines the base learners’ predictions for a final prediction. Combining classifiers with stacking has been shown to improve accuracy over best individual classifier in many cases [13][14][15][16]. Therefore, stacking is chosen for trying to improve the prediction performance of the artist chooser solution. In the process, different individual base classifiers are evaluated on the problem.

In this work, we explore different stacking approaches. We evaluate their performance on the artist chooser problem, as well as on eight additional datasets, and report the results. Our work contributes to the field of machine learning.

1.2 Problem

Wolpert, in his original paper on stacked generalization [3], described a few aspects of the technique as 'black art'. Those aspects have to do with how to best configure the stacking approach for maximum performance. The main issues are: 1) what level-0 generalizers (base learners) to use, 2) what type of attributes should be used to form level-1 (meta-level) input data and 3) what level-1 generalizer (meta learner) to use.

Regarding the first issue, it is well known that using a diverse set of base classifiers, with relatively uncorrelated prediction errors, results in good performance [9]. For the second one, there have been multiple researches focused on finding the best meta-features [17][18], where it has been shown that for a successful stacking approach it is best to use output class probabilities of the base learners, instead of single class predictions, as meta-level input data [13]. For the third problem of finding the best meta learner, we think there is still some research yet to be done.

Early on in stacking research Ting and Witten [13] showed that using multi-response linear regression as a meta learner to perform model combination provided the best results. This research was extended by Džeroski and Ženko [14], which showed that using multi-response model trees as a meta learner performed better than the linear regression model used in Ting and Witten’s study. In a later study, Doumpos and Zopounidis [15] compared seven different learning algorithms as meta learners, both linear and non-linear. Their results showed that logistic regression
performed the best as a meta learner - supporting Ting and Witten’s result that using a simple regression model to combine base learners’ predictions provides the best result. In a recent study, Lee [19] explored the appropriate combination of base learners and meta learners. The result showed a combination of a simple base learner (such as logistic regression) and a complex meta learner (such as a neural network) worked the best.

In machine learning competitions, top solutions often use stacking with Gradient Boosting Machine [20] / XGBoost [21] as a meta learner [22][23][24], as well as being used in some recent literature [25].

From covering the related work on meta learner comparisons, we find that the results are mixed. There is still a knowledge gap regarding the issue of choosing the best meta learner for stacked generalization. Choosing the right meta learner for a stacking solution is important in order to utilize the full strengths of each of the individual base learners and to combine their predictions in the best possible way for a final prediction. We think evaluating different stacking approaches, using linear and non-linear meta learners, on the artist chooser datasets, and other datasets, could both improve the artist chooser solution, as well as bring new and interesting knowledge to the field of machine learning.

1.3 Purpose

The purpose of the thesis is to evaluate and compare different stacking approaches and try to fill the knowledge gap addressed in the previous section. We compare non-linear model to a linear model as a stacking meta-level learner. We use different input meta-features: the base learners’ class probabilities and the original input features, and look at what effect they have on the different meta-level learners. Results for each stacking approach are presented and evaluated with regards to classification error rate. With this thesis we try to answer the following research question:

- How does a non-linear model compare to a linear model as a meta-level learner for stacked ensemble?

The purpose of the work is to evaluate different methods for choosing a correct artist for an incoming track/album ingested into the DRMC’s system, with the goal to improve the current artist chooser solution.

1.4 Goal

In order to try and answer our research question, the following things need to be achieved. First, we evaluate the current artist chooser solution, using Random Forest. After we have a baseline, we evaluate different individual models to see how they compare. Stacking is then used to combine the individual models, the
1.4. GOAL

base learners, to further improve the performance of the artist chooser solution. We conduct experiments to compare and evaluate four types of stacking approaches on a total of ten different datasets. The evaluation datasets are the two artist chooser datasets, as well as eight datasets from the UCI Machine Learning Repository [4]. The additional datasets were selected with the objective to have them as diverse as possible, in order for us to get an estimate of the general performance of the different stacking approaches. We use the ten datasets to evaluate stacking with a non-linear meta learner (gradient boosting machine) and with a linear meta learner (logistic regression) using two different types of input meta-features: 1) the base learners’ class probabilities and 2) a combination of the base learners’ class probabilities and the original input features. The results are analyzed in order to find the best performing stacking approach.

The goal of the work is to propose an improved solution to the artist chooser problem and report findings of how different individual models, and stacking approaches perform on the problem. Furthermore, the same stacking approaches are evaluated on eight additional datasets to see how well our solution generalizes.

1.4.1 Benefits

Comparing individual models to stacking, and comparing a non-linear meta learner to a linear meta learner on ten different datasets increases our understanding of the performance of stacking for binary classification. This contributes to the field of machine learning.

Reporting results of future importance and the performance of different models and stacking approaches can benefit anyone who faces a similar problem of choosing an artist or author entity from limited number of features. Having access to information about how different approaches perform, with regards to error rate and time complexity, can support ones who are choosing a solution for similar problems.

Finally, using the two datasets from the DRMC to evaluate different approaches to the artist chooser problem can result in new discoveries about the datasets and the problem. That information could benefit those working on the artist chooser solution and aid in future decisions regarding it.

1.4.2 Ethics

This study explores different models and approaches to stacking. It uses real-world data from a DRMC. The datasets do not include any information on users. It includes non-sensitive information about tracks/albums and things related to it, such as artists, composers, producers, label etc. Other datasets used (from the UCI Machine Learning Repository) are public and do not include any sensitive information.

When training the models for predicting artists, it is important to train on diverse examples of data. We need to be aware of and look into what kind of errors the models make. If, for example, we only train on data related to very popular
artists, the models might learn a bias regarding attributes that those artists have, resulting in tracks and albums potentially ending up at the wrong artists. This we need to be aware of.

1.4.3 Sustainability

Thousands of tracks/albums are delivered to the DRMC each day (many of them being updates). A part of those tracks/albums belong to artists with ambiguous names. Manually resolving these issues each day would require a lot of man-hours. Automating the process with machine learning techniques of selecting an artist eliminates the inefficiency of doing it manually and saves money for the company. The performance of the current artist chooser solution is however not always good enough, resulting in users/artists/labels complaining about incorrectly placed albums that the DRMC’s employees manually need to correct in their system. This is not ideal, as employees are spending precious time correcting these incorrectly placed tracks and albums. Improving the classifier’s performance for this problem results in more satisfied stakeholders and less man-hours spent on making corrections in the system.

There is also another important sustainability aspect of this problem that needs to be considered. Most machine learning based systems that train on large amount of data have high computational cost, resulting in high energy consumption. Sustainable data modeling \cite{26} is a form of data modeling technology that aims to introduce a new perspective for engineers, focusing on approaches that maximize learning accuracy with minimum computational cost. This is an approach we have in mind when conducting this study; a number of different, diverse models, as well as stacked ensembles, are evaluated on the artist chooser problem with regards to error rate and computational cost. For the company, an informed decision can be made on choosing a model/stacking setup for the problem that meets prediction performance requirements while minimizing computational costs. To reduce the model’s complexity even further, only the most important training features could be used. Comparing a non-linear meta-level learner to a linear meta-level learner also helps in this regard for the field of machine learning. Non-linear models have usually higher computational complexity than linear models. We study the performance differences of the meta learners too see if the change in accuracy is worth the increased computational cost.

1.5 Methodology

One of the first things that need to be considered when conducting a degree project is the choice of research methods and methodologies. The choice of a methodology comes first. Methodologies are processes that are followed throughout the entire research project and can be of either a qualitative or a quantitative kind. Quantitative research is about providing a phenomenon, by experiments and tests on large datasets, while qualitative research is about studying a phenomenon, or artifact,
to understand meaning and behaviors, and to create theories using smaller, often unstructured, datasets [27].

For this study, we are trying to answer the specific research question of how a non-linear model compares to a linear model as meta learner for stacked generalization. Wolpert and Macready’s *no-free lunch* theorem [28] states that no single approach can be better than another for all classification tasks. We can therefore not theoretically prove that one meta learner is always better than another for all classification tasks. We can however conduct an empirical study, where we compare performances of different machine learning models and stacking approaches using non-linear and linear meta learners, on different datasets. We find that it is most suitable for our study to collect and analyze the results using quantitative methods, therefore a quantitative research approach is chosen.

After having chosen a research methodology, we have set a direction for how to conduct the thesis project. For a successful thesis, it is important to choose methods that belong together and can be used together [27]. For the rest of this section we will be presenting philosophical assumptions, research methods and research approaches that can be used in research projects, and then choose methods that best fit for this quantitative research. Research strategies, data collection, data analysis and quality assurance are presented in Chapter 3.

### 1.5.1 Philosophical assumptions

A philosophical assumption is the standpoint for the project and steers the whole research. It describes what view the research will have and how valid the collected data is in the eyes of the researchers. The following are several core assumptions that have been described well by Håkansson [27]:

- **Positivism**: Assumes that the reality is objectively given; independent of observer and instruments. This assumption works well with quantitative research; for experiments and testing performances.

- **Realism**: Assumes that things in reality are known or of a perceived existence. The researchers work with understanding the collected data to develop knowledge. Works well for qualitative or interdisciplinary research.

- **Interpretivism**: Assumes that to understand a phenomenon, a deep and complex exploration needs to be done to discover what meaning people assign to it. Works well for qualitative research.

- **Criticalism**: Assumes that the reality cannot be given, that it is produced and reproduced by people. Works well for qualitative research, for example when learning about how culture affects computer usage.

For our work of answering the research question how different stacking approaches compare, *positivism* is chosen as the view and philosophical assumption.
We find positivism to be the most suitable point of view for the research, as the research question implies an empirical study and the use of a quantitative approach to data collection and data analysis to reach a conclusion.

1.5.2 Research methods

Research methods provide procedures to carry out and complete tasks that support the overall process of the research. The following are some of the most common research methods, described well by Håkansson [27]:

- **Experimental research**: Deals with variables, their relationships and causalities. Works well for quantitative research when investigating systems’ performances.

- **Non-experimental research**: Describes or predicts behaviour or opinions, such as users’ opinions on systems’ functionalities. Works well for qualitative research.

- **Descriptive research**: Describes characteristics, but not causes or occurrences, for a situation. Works well for both quantitative as well as qualitative research and can be used to find new characteristics and meaning in already existing data.

- **Analytical research**: Concerned with validating hypothesis using existing information; known facts. Works well for qualitative research.

- **Fundamental research**: Focuses on fundamental principles and testing theories. It can be used when studying phenomenon to get new insights, resulting in new innovations, principles and theories, while challenging existing ones. Works well for quantitative as well as qualitative research.

- **Applied research**: Often used on top of fundamental research, focused on answering specific questions or solving known problems. Works well for quantitative as well as qualitative research.

- **Conceptual research**: Can be used for investigating concepts, using for example literature reviews and historical research to develop new concepts or interpreting existing ones. Works well for qualitative research.

- **Empirical research**: Used to gain knowledge from experiences and observations, focuses on real people and situations to test predictions. Works well for qualitative research.

As has been touched upon, we are trying to answer the research question of how a non-linear meta-level learner compares to a linear meta-level learner for stacked generalization. The question implies the use of quantitative methods to collect and analyse the data. In order to reach a conclusion we need to: 1) collect measurable
data on the performances of the different stacking approaches and 2) analyse the data to see if there is a true difference between the approaches. Therefore, experimental research method is chosen for the study. The different meta-level learners and meta-features are the variables in our experiments, and the change in classification error rate on each of the evaluation datasets is measured and inspected when the variables are changed.

1.5.3 Research approaches

Research approaches are concerned with drawing conclusions from the collected data and establishing if hypothesis are true or not. There are two main approaches and one hybrid approach, described well by Håkansson [27]:

- **Inductive**: Approach commonly used in qualitative research. After enough data is collected, it is analyzed to gain an understanding of phenomenon.

- **Deductive**: Approach commonly used in quantitative research. It is used to test theories and hypotheses with large datasets.

- **Abductive**: A mixed approach that uses both inductive and deductive approaches to establish conclusions. It starts with an incomplete dataset and a hypothesis that best explains it, then tries to find a possible explanation using preconditions.

Wolpert [3] described finding the best meta-level learner for a stacked ensemble as a "black art" problem. He said that one must usually rely on prior knowledge, in order to try and make an intelligent guess on the best method for combining the predictions of the individual classifiers. In this study we compare different meta-level learners on various datasets, small and large. We research what stacking approaches perform the best on the artist chooser problem, as well as on the additional UCI Machine Learning Repository datasets, with regards to classification error rate. We use statistical tests to evaluate the significance of the results. From our results we can draw the conclusions which stacking approaches are likely to perform well, and which ones are less likely to perform well, for similar binary classification problems. The conclusions can serve as an useful heuristic for those who are implementing a stacked ensemble. Therefore, an abductive approach is chosen.

1.6 Delimitations

As the evaluation of the different stacking approaches is first done on the artist chooser problem, which is a binary classification problem, and then done on additional datasets to see how well the solution generalizes, all experiments are done on binary classification problems. This research therefore only looks at how a non-linear meta learner performs against a linear meta learner on datasets where the target variable can take one of two classes.
Only four stacking approaches, using one non-linear meta learner and one linear meta learner, are evaluated and compared. This is done to get a more clear comparison between a non-linear and a linear learner. There might be other models that perform better on our datasets, but are not evaluated in this research.

1.7 Outline (Disposition)

The remainder of the thesis is organized as follows. In Chapter 2, a background for the practice of using automatic methods for solving name disambiguation problems is given, along with a theoretic background on machine learning techniques, followed by a cover of related work. In Chapter 3, research strategies, data collection methods, data analysis and quality assurance used for conducting the project are covered. Chapter 4 describes the evaluation datasets, the designs of experiments and the data analysis techniques used. In Chapter 5, the results of the experiments are presented. Finally, Chapter 6 concludes the results and discusses future work.
Chapter 2

Background

This chapter covers theoretic background and related work for the thesis project. First, a general overview of name disambiguation problems is given along with a discussion on how machine learning techniques can be used to solve such problems. Theoretic background on machine learning is given, where a few common supervised machine learning models are described, followed by an overview over ensembles, there among: stacked generalization. Finally, related work is covered.

2.1 Name disambiguation

In the DRMC’s system, information about all the content on their service, i.e. the metadata for the music, is stored in a metadata database. All tracks and albums are linked to Artist entities, which are the performers. Artists in the DRMC’s system can share the same artist names, but they still all have unique, internal identifiers. There is however no universal standard system in place for identifying artists in the industry. This can cause problems when multiple sources deliver new tracks and albums to the DRMC, as the only way they have to identify the tracks and albums’ artists is by name. The DRMC then has a problem of mapping the incoming tracks and albums to the correct artist in their system (if the artist name is ambiguous). Generally, this problem is known as name disambiguation.

Name disambiguation problems are well known and are not restricted to a single context. A problem very similar to the artist chooser problem is author name disambiguation problem. A recent survey on automatic methods for solving the author name disambiguation problem [6] discusses two different machine learning techniques that can be used for author assignment: classification or clustering. Using the classification technique for this problem is effective and has been successful many times, however, it requires a large number of labeled training examples to be available. Clustering techniques can also be effective and training examples are not needed, but information about correct number of authors or author groups might be needed [6].

If a classification technique is used to solve the name disambiguation problem,
one has to be careful about the quality of the data labeled and used as training data. When creating training data out of, for example, a publication database or a catalogue of music, it is important to know that the data available might not all be correct. In the case of training data for artist name disambiguation, one has to be aware that the data might contain some noise; that is, that it might contain some examples of tracks or albums that are assigned to incorrect artist in the catalogue, but are considered correct as nobody has suggested an edit on them. Using a single predictor on an imbalanced or noisy dataset can result in the predictor being highly affected by it \cite{29}. Therefore, in these cases, combining classifiers would be an advantage. Having multiple classifiers trained on the same dataset, where their predictions are then combined; the final prediction should not be as affected by the noise or imbalance of the data.

### 2.2 Machine learning

Machine learning is a subfield of artificial intelligence. It is, as described by Arthur Samuel, the field of study that gives computers the ability to learn from data, without being explicitly programmed \cite{30}. This means that unlike traditional development of computer programs, where conditional statements are written as instructions for how the system should behave, the system learns from experience how best to solve the problem at hand. This approach has many advantages, for example, when explicitly writing conditions for programs becomes too complex to design and maintain, or when it is hard to map relationships between features.

There are different ways in how learning can be done by computers. The two main learning methods are supervised learning and unsupervised learning. In supervised learning, a model is trained on a set of inputs and outputs to learn a mapping between them. The model is then used to predict the outputs of unseen input instances. The two main task of supervised learning are regression, concerned with predicting a continuous numerical value from an input instance, and classification, concerned with assigning a label to an input instance. In unsupervised learning however, no outputs are given for inputs in the training process. This learning method is concerned with finding new knowledge in the data and to find structure in its inputs. A common technique in unsupervised learning is clustering, where a set of inputs are divided into groups based on their qualities.

Here we will focus on the classification task of supervised learning as it is the learning method we have chosen to implement for our problem. We describe a few common and popular learning algorithms for those tasks.

#### 2.2.1 Logistic regression

Logistic regression (LR) is a generalized linear model (GLM) used for classification. It works similar to linear regression, where the goal is to find a set of weights that pair with the input features to minimize a loss function. The difference is however, that in linear regression an error metrics such as least-squares is used to measure
2.2. MACHINE LEARNING

'closeness' between label and prediction and the outcome is continuous. Whereas, in logistic regression, a logistic loss is minimized and the outcome is categorical.

For binary classification, logistic regression models the probability of a given input belonging to class 1. The predicting probabilities are in the range 0 to 1. More formally, for input vector \( \mathbf{x} \) and weight vector \( \mathbf{w} \) our net input function is 
\[
z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \mathbf{w}^T \mathbf{x}.
\]
The probability that an input \( \mathbf{x} \) belongs to class 1 is written in Equation 2.1:
\[
P(Y = 1 | \mathbf{x}) = \frac{1}{1 + e^{-z}}
\] (2.1)

This function is known as the logistic sigmoid function. It squashes \( z \) into the range of 0 to 1, interpreted as probabilities. The probability of an input vector belonging to class 0 is then simply 
\[
P(Y = 0 | \mathbf{x}) = 1 - P(Y = 1 | \mathbf{x}).
\]
In Figure 2.1 the logistic sigmoid function has been plotted. If \( \sigma(z) \geq 0.5 \), then the probability of the input sample belonging to class 1 is higher than 50% and we classify it to class 1.

![Figure 2.1. Logistic sigmoid function.](image)

From Equation 2.1, we can define the inverse of the logistic function, as the logit (or log odds) function, written in Equation 2.2:
\[
\ln \left( \frac{p(Y = 1 | \mathbf{x})}{1 - p(Y = 1 | \mathbf{x})} \right) = z = \mathbf{w}^T \mathbf{x}
\] (2.2)

This shows us that the model is a linear combination of the inputs, but still relates to the log-odds of our default class 1. The optimization problem is finding
the best values of the weight vector $w$. That is done using maximum-likelihood estimation.

In Figure 2.2 we can see an example of how a logistic regression model splits the input space linearly. It creates a decision boundary and classifies all points left of the line to the red class and all points right of the line to the blue class. The further away a point is from the line, the more confident the model is classifying it correctly.

![Figure 2.2. An example of a decision boundary for a logistic regression model.](image)

### 2.2.2 Naive Bayes

Naive Bayes (NB) methods are a set of simple classifiers based on Bayes’ theorem. They all make the "naive" assumption that every pair of feature values are independent of each other. A NB classifier is probability based. Given a feature vector $x_1, ..., x_n$, where $n$ is the number of features, the probability of it belonging to class $y$ can be computed according to Bayes’ theorem (Equation 2.3).

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n)}{P(x_1, ..., x_n)} \tag{2.3}$$

Now making use of the naive assumption that each feature is independent of the value of any other feature, given the class variable, and that $P(x_1, ..., x_n)$ is constant, we can use the following classification rule (Equation 2.4) to find the most probable class $\hat{y}$ for the given input feature vector.

$$\hat{y} = \text{argmax}_y P(y) \prod_{i=1}^n P(x_i|y) \tag{2.4}$$
Despite the naive assumptions on which the algorithm is based, it has still been shown to have surprisingly good performance [31]. There are several variations of the NB classifier available. They mainly differ by the assumptions they make regarding the distribution of the feature vectors. In this thesis we use the Gaussian Naive Bayes classifier as one of the base classifiers.

2.2.3 Decision trees

Decision trees (DT) are a tree based algorithm that can be used for both regression and classification predictions. Here, we focus on classification trees, where the target variable is discrete, compared to regression trees, where the target variable is continuous.

A decision tree represents sequence of rules that it goes through to classify an input instance. Each branch in the tree represents a if-then-else decision rule based on a feature. The leafs (terminal nodes) determine the class that the input instance belongs to. Figure 2.3 shows an example of a decision tree.

![Decision Tree Diagram](image)

**Figure 2.3.** An illustrative figure of a decision tree. $F_i$ denotes the feature number $i$. When an input instance is being classified, it starts at the top and follows a path by asking the questions in the nodes. When the input instance has reached a leaf, it is classified according to the leaf’s class, i.e. either assign or not assign in this case.

A decision tree is built at training time, starting from the top (root) and grows downwards, using recursive binary splitting [5]. At the root, all training examples are available. The algorithm then tries to find the best split of the data, by selecting a feature, to minimize a classification error. All data points whose value for the selected feature is less than or equal to the best split point, decided by the algorithm, are assigned to the left branch. The rest of the data points, whose value for the selected feature is greater than the split, are assigned to the right branch. This process is done recursively, but now the left node and the right node have fewer data points to consider. These nodes could either split the data points further, using features that have not been used before, or do a prediction - making them terminal nodes (leafs). The most commonly occurring class of the training data points, at a terminal node, will be the terminal node’s predicted class. The classification error rate is simply the fraction of the training data points at the nodes that do not belong to the most common class. In practice however, classification error is not sufficiently sensitive for tree growing, so evaluation metrics such as the Gini index or entropy are used to determine the quality of the split [5]. The algorithm iteratively
continues to split the data until: 1) every node has a "pure set" of data points (all belong to the same class), 2) all features have been used or 3) when it has reached a pre-determined maximum depth.

One of the disadvantages of decision trees is that they can overfit. Too many rules fit on the training data won’t always generalize well and predictions will become less accurate. Overfitting can be handled, for example, by pruning, where parts of the tree, that don’t make significant improvements to the overall accuracy, are removed. Other common methods for reducing overfitting is to use an ensemble of decision trees, such as Random Forest (see Section 2.3.1).

The advantages of using decision trees are for example: 1) they are easy to interpret and 2) they can capture non-linear decision boundaries.

2.2.4 Artificial neural network

Artificial neural networks (ANN) are highly sophisticated learning models capable of modeling complex non-linear functions. They have gained increased popularity in recent years due to improvements on the technique along with more data and processing power becoming available. Neural networks are inspired by the biological neural networks of the human brain. It was Frank Rosenblatt’s perceptron, from 1958, that was the first algorithmically described neural network [32]. The perceptron is built around a single neuron - modeled earlier by McCulloch and Pitts - and is able to perform a pattern classification with two classes. It takes several binary inputs and produces a single binary output. For the perceptron to work properly however, the two output classes have to be linearly separable [32].

In 1969, Minsky and Papert published a book that criticized the perceptron. They pointed out that the limitations of the single-layer perceptron was its inability to solve classification problems which are not linearly separable - such as the XOR problem. They also gave their concerns regarding the computational capabilities of the approach, stating that it was not a viable method [32]. The publication of the book led to decreased interest in neural networks for a long time.

Later, the introduction of the back-propagation algorithm renewed interest in neural networks. The algorithm effectively solved the XOR problem and improved the training performance.

Nowadays, there exist many different types of neural network architectures, some of which we will cover briefly. A neural network consists of layers, that are of three different kinds:

1. The **input** layer is responsible for taking information into the network and passing it forward. Each node in the layer corresponds to a different input feature.

2. The **hidden** layer(s) perform computations and transfer information from the input nodes to the output nodes. The nodes in this layer are neurons, which have an activation function for doing computation on the data.
3. The **output** layer also has neurons performing computations. Additionally to
that it provides the response from the network.

The simplest and most common type of neural networks is the feedforward neural
network (FFNN). A representative class of such a network is Multilayer Perceptrons
(MLP), a network that has one or more hidden layers of neurons [32]. Figure 2.4
shows an example of MLP network that has two input nodes, one hidden layer with
three neurons and one output node.

![Graph of a multilayer perceptron network with one hidden layer.](image)

In a feedforward neural network, the information flows in only one direction:
forward. The connection between the nodes in adjacent layers have weights. The
network is able to learn, using the back-propagation algorithm, by comparing the
outcome of the network to the expected outcome. It propagates the error back to
previous layers, where it is noted and the corresponding weights are updated to
try to minimize the overall error. This process is iterated until the error rate has
dropped below a certain threshold, or if a stopping criterion has been reached.

The purpose of the hidden neurons is to act as feature detectors. They are able
to gradually discover important features that characterize the input data. They do
so by performing non-linear transformation on the input data using its activation
function. A neuron’s activation function can be for example the *sigmoid* function,
as described in Section 2.2.1. Other popular activation functions are, for example,
*rectified linear unit* (ReLU) and *exponential linear unit* (ELU). The result of the
input transformation is a new space called the feature space, where classes can be
more easily separated [32]. Figure 2.5 shows an example of how a MLP can create
a non-linear decision boundary to classify two kinds of input data.

Other notable types of neural network architectures are convolutional neural net-
work (CNN), which work well with computer vision and grid-like data, and recurrent
neural network (RNN), which work well with sequence data, such as predicting the
next word in a sentence.
2.3 Ensemble methods

A wide variety of learning algorithms are available for solving machine learning tasks. Each algorithm has its own advantages, but still, no single approach can claim to be better than another for all classification tasks. This has been shown by the no-free-lunch theorem [28]. Different classifiers often make different errors on predictions. Combining the prediction of multiple diverse classifiers, all trained to solve the same problem, can in many cases improve predictions over any single classifier by amplifying correct decisions of the classifiers and cancel out the incorrect decisions [9].

Systems concerned with combining the predictions of multiple classifiers to improve prediction performance are called ensemble based systems. These systems emulate the common human behaviour of consulting a number of experts on a topic before making a decision related to it [9].

The task of constructing an ensemble based system is made up of two parts: 1) building a set of diverse classifiers and 2) combine the outputs of the individual classifiers for a final prediction. There are several possible ways to do this. One approach can be to use a set of homogeneous classifiers that use different samples of the training data or are configured in different ways. Their outputs can then be used as votes for a final prediction. Another approach is to combine the predictions of a set of heterogeneous classifiers. A meta learner is then used to combine their predictions. Here we will describe in more detail the most common ensemble methods and what techniques they use to build and combine classifiers.
2.3. ENSEMBLE METHODS

2.3.1 Bagging

Bagging [10] is a simple and effective ensemble technique that partitions the training set and trains base classifiers on different samples of it. Using diverse base classifiers improves the overall prediction performance of an ensemble. If the classifiers are diverse, it means that they will make errors in different ways. If the classifiers are also accurate (at least slightly better than a random guess), then these errors should be eliminated when the classifiers vote on the final prediction. Let’s take for example an ensemble with three diverse base classifiers, one classifier could make an error on a particular prediction, but if the other two make the correct prediction, they have the majority vote for the final prediction and the error of the single classifier will be canceled out. One way of having diverse set of base classifiers is to train them on different non-overlapping subsets of the training data, but since training data is often of a limited size, bagging solves this in a different way.

Bagging applies bootstrap sampling to create $k$ training sets for $k$ different classifiers. All training sets contain the same amount of training examples as the original training set. The sampling is done with replacement, which means that in the training sets, some original examples might appear more than once, while others might not be present. Using this technique is particularly useful for extending a training set of a limited size. This however means that many training sets could share a high portion of the same training examples. In order to ensure diversity of the base classifiers, using unstable classifiers, such as decision trees or neural networks, are recommended, as it is possible to control their instability with configuration parameters [9].

Random Forest

Random Forest [8] is a variation of the bagging algorithm. It is an ensemble constructed from a large number of decision trees. Practical decision tree learning algorithms are often based on the greedy approach [5], resulting in the algorithm tending to select the same sequence of strong features to build the trees. This leads to decreased diversity among the trees in the ensemble and therefore decreased accuracy. The random forest learning algorithm differs from the bagging algorithm in how feature selection is done. It tries to decrease correlation among the trees by selecting a random subset of the features for each split. The trees then use the best features from the subset of features available to them. For a classification problem with $p$ features, Breiman [8] used $\log_2 p + 1$ features for each split in his evaluation of the algorithm. Typically, in practice, $\lfloor \sqrt{p} \rfloor$ number of features are used for each split. This is for example the default value of the max number of features for a split in scikit-learn’s implementation of the random forest classifier [33].

2.3.2 Boosting

Similar to bagging, boosting [11] creates an ensemble of classifiers by re-sampling the training data. The approach is completely different however. While bagging trains
each classifier on splits in parallel, boosting trains classifiers in a sequential order, each one improving the previous one. The goal is not to minimize the classification error in each tree, but to further reduce the classification error of the ensemble. The learning algorithm is able to convert weak learners to strong learners, where a weak learner is only slightly better than a random guess and a strong learner can correctly classify all but an arbitrarily small fraction of the instances [11]. This is generally done by re-weighting the training examples according to the result from the previous classifier trained on the data. The general boosting algorithm is not iterative, so it is not able to fully adapt to the weak classifiers. More advanced boosting algorithms have been developed with this in mind and here we will described two of them.

**AdaBoost**

AdaBoost [34] is the most well known and influential boosting algorithm. It improves on the original algorithm via an iterative process, making it adaptive to the weak classifiers. The algorithm works as follows. First, a single classifier is trained with a random subset of the training data. Then, a second classifier is trained on data from the previous classifier, where examples that were misclassified get higher weight than the examples that were correctly classified. This means that future classifiers can focus more on instances that are difficult to classify. The number of weak classifiers used in the ensemble is a configuration issue. A final prediction is then done by weighted majority voting, where classifiers that performed well during training get higher voting weights.

**Gradient boosting**

Like other boosting algorithms, gradient boosting (often called gradient boosting machine) produces a prediction model made up of an ensemble of weak classifiers, typically decision trees. What is special about the algorithm is that it combines gradient descent and boosting. Unlike in AdaBoost, where the algorithm is optimized by focusing on high-weight data points, in gradient boosting, it is optimized around a suitable cost function. The cost function needs to be differentiable. We use partial derivatives of the cost function to measure the gradient. We can then use the gradient to find the descent direction and update our weights towards finding the minimum loss of the cost function. So after calculating the loss for a weak learner, the gradient descent procedure is performed, and a new tree is then added to follow the gradient and further reduce the loss until a halting point is reached or until we have added the predefined number of weak learners (trees).

**2.3.3 Stacked generalization**

Stacked generalization (stacking) was first introduced by Wolpert in 1992 [3]. Unlike other ensemble based methods where predictions of a number of homogeneous learning algorithms are combined by voting, stacking uses a second level learning
2.3. ENSEMBLE METHODS

algorithm to combine the predictions of a number of heterogeneous learning algorithms. If the base learners have been selected from a pool of diverse classifiers, they should have somewhat different decision boundaries and are therefore able to learn different mappings. This also means that diverse base learners make errors on different instances, and by combining their predictions, these errors can be cancelled out if the other learners don’t make them as well. Overall, combining classifiers with stacking has been shown to improve accuracy over best individual classifier in many cases [13][14].

Generally, the approach to stacking can be divided into two phases. In the first phase, a set of base learners, $L_1, L_2, \ldots, L_T$ are created and trained. In the second phase, a meta learner, $L_{T+1}$, is trained on the output hypothesis, $h_1, h_2, \ldots, h_T$, of the base learners and the corresponding true labels. That way, it learns how to best combine the predictions of the base learners for a final prediction. The meta learner is trained using a leave-one-out or a cross-validation (CV) procedure. Using K-fold CV, a dataset $S$ is split into K-parts. The base learners are then trained on all but one part, which is held out as a validation set. The output predictions from the base learners on the validation set are combined in a vector, along with the correct label for each example, which the meta learner is then trained on. This is done for all $K$ folds of the dataset, that is, until all parts have once been used as a validation set. Once the meta learner has been trained, the base learners are re-trained on the entire dataset for an improved accuracy. When a prediction is made from input data $x$, first all base learners make their predictions from the input. The base learners’ output hypothesis (predictions) are then used as input to the meta learner. With that information, the meta learner makes a final prediction. Figure 2.6 illustrates the stacked generalization approach.

Figure 2.6. An illustrative figure of stacked generalization. The level-1 generalizer combines the predictions of the level-0 generalizers for a final prediction.
The most important issues in stacking are: 1) what type of attributes to use to form the level-1 (meta-level) input data and 2) what level-0 generalizers (base learners) to use and what level-1 generalizer (meta learner) to use. In the next section we will cover some of the work that has been done to try and answer these questions.

2.4 Related work

Ever since stacking was introduced, the approach has been used in many application domains for improving prediction performance. Multiple classifier systems in general have been receiving increased attention in recent years, where the growth in number of publications is rising fast [35]. The approach is also very popular and effective in machine learning competitions, where for example the top teams in the Netflix competition utilized a blend of multiple classifiers [36][37].

Wolpert [3] initially described the main aspects of stacking as 'black art', saying that there are currently no hard and fast rules to what base learners to use, what meta-level learner to use and how to form the meta-level input space. Furthermore, he suggested that in practice, one must be content to rely on prior knowledge when deciding on how to best configure this. A number of papers have been published since then that try to find the best stacking approach. Here we cover some of that work.

The problem of finding the best meta-features revolves around finding what form of predictions, provided by the base learners, are the most effective when combined by the meta-level learner for a final prediction. These forms of predictions can be, for example, single class predictions or class probabilities. Merz [17] proposed with his stacking method, SCANN, to use correspondence analysis to detect correlations between the predictions of the base learners and then to transform the meta-level data to remove these correlations. Ting and Witten [13] proposed to use class probabilities rather than single class prediction for the meta-level data. They found this approach to be crucial for the success of stacking in classification tasks. Seewald [18] looked at a weakness in stacking for multi-class classification when using class probabilities and Multi-response Linear Regression (MLR) as a meta learner. Seewald argued that this weakness was due to the high dimensionality of the meta-data and proposed a solution that reduces the dimensionality by a factor equal to the number of classes in the dataset. Džeroski and Ženko [14] extended the set of meta-level features used by Ting and Witten [13], additionally to using the class probabilities, they propose using the probability distributions multiplied by the maximum probability and the entropies of the probability distributions as meta-features. These attributes try to capture the certainty of the class predictions more explicitly. They showed however that this only provides limited advantages over using class probabilities alone. Bao et al. [38] stacked multiple recommendation engines with additional meta-features. In their work, the input features to the meta-level learner are the predictions from the component engines combined with
2.4. RELATED WORK

runtime metrics that represent properties of the input users/items. Experimental results showed that their system outperformed each single component engine, indicating that including additional meta-features along with the predictions of the base learners can increase the performance of a stacking solution.

For the problem of selecting base learners, using a combination of diverse learners is a good approach. This is in fact the cornerstone of ensemble systems: create many classifiers and combine their outputs such that the final prediction improves the performance of a single classifier [9]. When using a set of diverse classifiers, they should learn different mappings of the data and make different errors. Strategically combining their predictions can reduce the total prediction errors. In stacking, it is common to use a set of different classifiers as base learners, such as logistic regression, support vector machine, decision trees and artificial neural networks.

How to best combine the predictions of the base learners is another problem. Ting and Witten [13] compared the effect of four different learning algorithms as the meta-level learner: C4.5, a decision tree learning algorithm; IB1, a variant of a lazy learning algorithm; NB, a Naive Bayesian classifier; and MLR, a multi-response linear regression algorithm. Among these algorithms, their experiments showed that only the MLR algorithm was suitable as a meta-level learner. Džeroski and Ženko [14] extended the study by Ting and Witten, where they replaced the MLR algorithm with a multi-response model trees as a meta-level learner. Their experiments showed that stacking with multi-response model trees performs better than stacking with multi-response linear regression. Doumpos and Zopounidis [15] compared different stacking approaches for credit risk assessment. They compared the effect of seven different learning algorithms as the meta-level learner: linear discriminant analysis, quadric discriminant analysis, logistic regression, probabilistic neural network, classification and regression trees, nearest neighbors and support vector machines. Their experiments showed that logistic regression performed the best, supporting Ting and Witten’s result that using a simple regression model as a meta-level learner provides the best result. Lee [19] used stacking to predict depression among the elderly. The study aimed to explore the appropriate combination of base-level and meta-level classifiers, where one classifier was used as a base-level learner and a different classifier as a meta-level learner. A pool of five classifiers were used for the study: logistic regression (LR), decision tree (DT), neural network (NN), support vector machine (SVM) and naive bayes network (NBN). The results were that a combination of a simple base-level learner (LR or DT) and a complex meta-level learner (NBN, NN, SVM) showed the best performance. The delimitations of the study are that only one type of a base-level learners was used at a time, instead of using a combination of different types of base-level learners as is commonly done in stacking.

From covering related work that tries to address the issue of finding the best meta-level learner, we find that the results are mixed. More work can be done on exploring this particular issue to address the knowledge gap in the field.
Chapter 3

Methods

This chapter covers methodologies and methods used for conducting the degree project. As introduced in Section 1.5, a quantitative research approach is chosen for this study. In order to answer the research question of how a non-linear model compares to a linear model as a meta learner for stacked generalization, we study empirically the different stacking approaches to see how they compare to each other, and how they compare to the best individual base learners, over ten evaluation datasets. For reaching a conclusion, we need to collect measurable data from our comparisons and analyse them accordingly.

In the following sections we will present research strategies, data collection, data analysis and quality assurance methods that can be used in a research project. After having covered that, we select the best fit methods to use for our research. We motivate our choice of methods used for answering the research question, and we discuss how the methods are applied in the degree project.

3.1 Research strategies

A research strategy is an overall plan for conducting a research. It is a guideline which includes organizing, planning, designing and conducting a research. The following are the most common research strategies, that have been described well by Håkansson [27]:

- **Experimental research**: Concerned with controlling all factors that may affect results and provides cause-and-effect relationship between variables. Analysis are done using statistics. Works well with experiments on large datasets.

- **Ex post facto research**: Similar to experimental research, but is carried out after data has been collected. It searches back in time to find possible causal factors. Can be used for example to study behaviour.

- **Surveys**: A descriptive research method. Examines frequency and relationships between variables. Possible to collect data over a period of time or at a


single point in time.

- **Case study**: Investigates a phenomenon, in a real life context, using multiple sources of information. Works well with both quantitative and qualitative data.

- **Action research**: Concerned with improving the way people solve problems. Communities or settings are studied where data is limited. Actions are taken, observed, evaluated and reflected. Works best with qualitative methods.

- **Exploratory research**: Identifies key issues and variables to define objectives by exploring the possibility to obtain as many variable relationships as possible. Uses qualitative data.

- **Grounded theory**: Systematically collects and analyses data and seeks to develop a theory on it.

- **Ethnography**: Studies people and culture. Seeks to place a phenomena in their cultural context.

In our study, we are trying to answer the question how different stacking approaches perform when predicting on real-world classification tasks. We use ten real-world datasets that we evaluate the different stacking approaches on. We want to control all factors of the study. We want to measure what effect it has on the classification error rate on the datasets, when we apply different meta-level learners for the stacking approach. Furthermore, we want to measure what effect using different input meta-features: the base learners’ class probabilities, or a combination of the base learners’ class probabilities and the original input features, has on the classification error rate when using the different meta learners. We start with the null hypothesis that there is no statistical difference between the error rates on the datasets using different stacking approaches. We then perform significance tests on our results to see if the null hypothesis can be rejected. Therefore, we chose to follow the *experimental* research strategy, as it is the most suitable one when conducting experiments and performing statistical analysis on the results.

### 3.2 Data collection

In order to evaluate the different stacking approaches, we first need to choose an appropriate data collection method for how to collect results. The following are the most common data collection methods, that have been described well by Håkansson [27]:

- **Experiments**: Collects datasets for variables being researched. Often large amount of data collected.

- **Questionnaire**: Collects data through questions.
3.3. DATA ANALYSIS

- **Case study**: Focuses on a small number of samples or participants to get a deeper understanding of the subject.
- **Observations**: Concerned with observing behaviour for certain situations and culture.
- **Interviews**: Direct approach for getting a deeper understanding of the subject, capturing participant’s point of view.
- **Language and text**: Interpret meaning of conversations and texts.

To answer the question of how different classifiers and stacking approaches perform on classification tasks, we need to collect measurable data of their performance that we can then compare. We choose to conduct experiments to do the data collection. Each approach is applied for predicting on ten classification tasks, and the performance metrics: classification error rate and prediction time, are collected.

### 3.3 Data analysis

To evaluate and compare the performances of the stacking approaches, we need to analyse the collected data from our experiments. The process of data analysis includes inspecting, cleaning, transforming and modelling the data. The following are the most common data analysis methods, that have been described well by Håkansson [27]:

- **Statistics**: Use statistical methods to analyse data and evaluate significance of the results.
- **Computational mathematics**: Calculates numerical methods, models and simulates.
- **Coding**: Turns qualitative data into quantitative data. Coding is applied to name and label data, numerate and apply statistics on.
- **Analytic induction/grounded theory**: Iteratively collect and analyse data until a valid theory has been formed.
- **Narrative analysis**: Concerned with analysing text, signs and symbols to support traceability in requirements.

For analysing the collected data from our experiments, statistical methods are chosen. Classification error rates of the individual classifiers and stacking approaches are compared and ranked using statistical tests. The outcome of the statistical test determines if the null hypothesis (stating that there is no statistical difference between the performance of the different approaches) can be rejected or not. If the null hypothesis is rejected, additional statistical tests are used for measuring between which approaches there is a statistical difference.
3.4 Quality assurance

The final thing necessary to think about when carrying out a research project is quality assurance, that is, how the work can be validated and verified. For a quantitative research, the following things, as described by Håkansson [27], need to be addressed:

- **Validity**: To make sure that the results are valid, that the test instruments are sound and actually measuring what they are expected to do.
- **Reliability**: That the measurements are stable and consistent.
- **Replicability**: The possibility that the same results can be reached for the same research by another researcher.

To ensure validity, reliability and replicability of the research, a detailed description of design, settings, metrics and evaluation techniques used, are give in Chapter 4. For the machine learning models used and evaluated, we state their configuration settings. We describe how the different stacking approaches being evaluated are designed so they can be re-implemented by other researchers. We describe the metrics used for evaluating the results, as well as the significance tests. The evaluation datasets used to train and test the classification algorithms are described. Finally, the tools and software packages used for conducting the experiments are listed in Appendix A.
Chapter 4

Design of Experiments

This chapter covers how the design of experiments are performed and how the results are analysed. In Section 4.1 we describe in detail the two main datasets, the artist chooser’s album and track datasets, used for evaluating the performance of the different stacking approaches. Additionally to those two datasets, we describe the eight datasets from the UCI Machine Learning Repository [4] that are used to see how well our results from the DRMC’s artist chooser use case generalize to other datasets of different kinds. In Section 4.2 we describe the different stacking approaches that are evaluated. In Section 4.3 we cover how the experiments are conducted and how the individual classifiers and stacking approaches are evaluated. Finally, in Section 4.4 we describe the statistical techniques used to evaluate the significance of the results from our experiments.

4.1 Datasets

We use in total ten different datasets to evaluate the performance of the different stacking approaches. The datasets are diverse, of different sizes and have different combination of features. They are all binary-classification datasets and relatively well balanced.

4.1.1 Artist chooser datasets

For the artist chooser problem, there are two similar datasets used for training two classifiers: album and track. These classifiers are used for predicting the correct album artists and track artists for albums and tracks that are ingested into the DRMC’s system. For example, when a new album by the artist "John" gets delivered, the album classifier is used to determine which Artist/s the album should be assigned to in the system. The track classifier is used to determine which Artist/s each track on the album should be assigned to. The only difference between the two datasets are they each have one feature the other one does not have, features specifically related to the album or the track/s. In Table 4.1 the properties of the
two artist chooser (AC) datasets are listed. A random sample of approximately one million examples from each of the two datasets are used for our experiments.

Table 4.1. The artist chooser datasets. The properties listed are: number of classes, number of instances, number of features and their types (binary/discrete/continuous), and the probability of the majority class.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Classes</th>
<th>Instances</th>
<th>Features</th>
<th>B/D/C</th>
<th>Majority class</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC - Album</td>
<td>2</td>
<td>1,000,005</td>
<td>47</td>
<td>13/3/31</td>
<td>0.64</td>
</tr>
<tr>
<td>AC - Track</td>
<td>2</td>
<td>1,000,008</td>
<td>47</td>
<td>13/3/31</td>
<td>0.66</td>
</tr>
</tbody>
</table>

In order to get a better understanding of the two datasets we do exploratory analysis on them. We use one of the base learners, a Random Forest model, to analyse the importance of each of the 47 features. The model is trained on the whole dataset and information about the relative importance of every feature, as used by the model, is extracted from it. Figure 4.1 shows the top 15 features of the AC - Album dataset. From it we can see that information about the album’s licensor, label and feed weight the most when it comes to determine a likely artist for an album. This indicates that if there is already an album by the candidate 'John' in the system from the same licensor, label and/or feed as the incoming album being classified, then it is likely to be the correct candidate artist, as from this information, the same licensor/label/feed doesn’t seem to have multiple artists with the exact same artist name.

Figure 4.1. The top ranked features of the artist chooser’s album data set. The graph shows the relative importance of all features of the dataset that rank higher than 1% in importance.
4.1. DATASETS

We trained multiple Random Forest models, using different seeds, on the whole album dataset and compared the top ranking features extracted from the models. In all cases the licensor/label/feed were the most important features. Other important features were: the ISRC country code of the album and if it matched the candidate’s most common country code; the album and candidate’s source match; the album year compared to the candidate’s mean year active; and features related to the artist name length. The exact order and importance of these top features varied a little bit between models. That can be explained by the correlation between some of these features. There are for example a few highly correlated features related to the artist name length or the ISRC country codes. This results in the model often only picks one of the correlated features, which then becomes highly important, while the other correlated features are not ranked as high, as they will not improve the performance of the model as much after that. Still, including all of these correlated features from the dataset improves the overall accuracy of the model, compared to excluding some of them completely. Figure 4.2 shows a correlation heatmap for the 47 features of the AC - Album dataset. It shows that overall, the features are relatively uncorrelated.

Figure 4.2. A feature correlation heatmap for all 47 features of the artist chooser’s album dataset. Feature names have been replaced with feature numbers.
4.1.2 UCI Machine Learning Repository datasets

In order to see how well our results from the artist chooser use case generalize, we selected eight diverse datasets from the UCI Machine Learning Repository [4] to use as well for our experiments. All these datasets have been used before in comparative studies, such as Džeroski and Ženko’s [14]. The eight datasets were selected with the objective to have them as diverse as possible. That way we can get a better picture of the general performance of the different stacking approaches over multiple, different datasets. The only required qualities the chosen UCI datasets needed to have were: 1) have only two target classes (binary classification dataset) and 2) be relatively well balanced. Otherwise, we tried to select datasets of various sizes, with different number of features and feature type combinations. A python wrapper from the Penn Machine Learning Benchmarks [39] was used to retrieve the UCI datasets in our experiments. In Table 4.2 the properties of the eight datasets are listed.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Classes</th>
<th>Instances</th>
<th>Features</th>
<th>B/D/C</th>
<th>Majority class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>2</td>
<td>48,842</td>
<td>14</td>
<td>1/7/6</td>
<td>0.76</td>
</tr>
<tr>
<td>Australian</td>
<td>2</td>
<td>690</td>
<td>14</td>
<td>4/4/6</td>
<td>0.56</td>
</tr>
<tr>
<td>Breast-W</td>
<td>2</td>
<td>699</td>
<td>9</td>
<td>0/9/0</td>
<td>0.66</td>
</tr>
<tr>
<td>Heart-C</td>
<td>2</td>
<td>303</td>
<td>13</td>
<td>3/5/5</td>
<td>0.54</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>2</td>
<td>351</td>
<td>34</td>
<td>0/0/34</td>
<td>0.64</td>
</tr>
<tr>
<td>MAGIC</td>
<td>2</td>
<td>19,020</td>
<td>10</td>
<td>0/0/10</td>
<td>0.65</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>2</td>
<td>958</td>
<td>9</td>
<td>0/9/0</td>
<td>0.65</td>
</tr>
<tr>
<td>Vote</td>
<td>2</td>
<td>435</td>
<td>16</td>
<td>0/16/0</td>
<td>0.61</td>
</tr>
</tbody>
</table>

4.2 Stacking setup

In this section we describe the design of our different stacking approaches. In Section 4.2.1 we describe the five base-level learners that are used for all the stacking approaches. In Section 4.2.2 we describe the four different schemes used for combining the base-level learners. The schemes consist of two different meta-level learners, both using two different types of meta-features. In Section 4.2.3 we describe the overall setup of the stacking approaches. Finally, in Section 4.2.4 we describe how we build and train the stacking approaches.
4.2. STACKING SETUP

4.2.1 Base-level learners

Having a set of diverse base-level learners, as discussed in Section 2.3.3, is one of the key things for a successful stacking solution. With that in mind, we select five different classifiers that serve as base-level learners for all our stacking approaches. We select a probabilistic classifier, a linear classifier, a variation of a boosting ensemble classifier, a variation of a bagging ensemble classifier and an artificial neural network classifier. We decide that using five base-level learners is a good and reasonable number for our stacking approaches. Comparative studies have used between three and seven base-level learners [13][14][15]. The five classifiers that serve as base-level learners are:

- **NB**: Gaussian Naive Bayes (Section 2.2.2)
- **LR**: Logistic Regression (Section 2.2.1)
- **GBM**: Gradient Boosting Machine (Section 2.3.2)
- **RF**: Random Forest (Section 2.3.1)
- **ANN**: Artificial neural network (Section 2.2.4)

All classifiers are used with their default parameter settings as implemented by scikit-learn [40]. That is done to get as general results as possible from our experiments over the different evaluation datasets. In our experiments, in addition to evaluating the different stacking approaches, we evaluate the performance of every single one of the base-level learners to find the best single performing model for each dataset. We use these results and compare the best models with the performance of the stacking approaches.

4.2.2 Meta-level learners

The main purpose of the thesis project is to evaluate and compare different meta-level learners for a stacking solution. More specifically, we want to evaluate how a non-linear meta-level learner performs compared to a linear meta-level learner. We choose logistic regression to be our representative linear meta learner. Logistic regression is a popular linear classifier and has been used as meta learner in comparative studies with good results [15][19]. We choose gradient boosting machine to be our representative non-linear meta learner. Gradient boosting machine is a popular ensemble classifier and is often used as a meta learner in machine learning competitions [22][23][24] and has been used as well to combine classifiers in recent literature with good results [25].

We want to look at the effects on stacking performance when using different types of meta-features with the different types of meta-level learners. We conduct experiments and evaluate the two meta learners using the base learners’ class probabilities as meta-features, as well as using a combination of the base learners’ class
probabilities and the original input features as meta-features. We evaluate in total four different schemes for combining classifiers:

- **SLR1**: Stacking with logistic regression (Section 2.2.1) using the base learners’ class probabilities.
- **SLR2**: Stacking with logistic regression (Section 2.2.1) using a combination of the base learners’ class probabilities and the original features.
- **SGBM1**: Stacking with gradient boosting machine (Section 2.3.2) using the base learners’ class probabilities.
- **SGBM2**: Stacking with gradient boosting machine (Section 2.3.2) using a combination of the base learners’ class probabilities and the original features.

As with the base-level classifiers, all meta-level classifiers are used with their default parameter settings to get as general results as possible over the different evaluation datasets.

### 4.2.3 Setup

The setup of our stacking approaches can be seen in Figure 4.3. When an input instance is classified, the base learners first make their predictions. Their predictions, and in some cases the original feature as well (denoted by the dotted line), serve as meta-features for the meta learner. The meta learner then makes a final prediction.

![Figure 4.3. The setup of our stacking solutions.](image-url)
4.2.4 Building and training the stacking approaches

The stacking approaches are built and trained using 10-fold CV. It means that the data used to train the stacking approaches is split into ten parts, where nine parts serve as a training set, and one part serves as a validation set. The base learners are trained on the training set and then make their class probability predictions on the validation set. Their predictions for each example in the validation set are gathered together, along with the correct labels for the examples. This is repeated ten times, where different parts of the data serve as a validation set, or until the base learners have made their predictions on all parts of the data. The predictions of the base learners on the data are used as input meta-features to the meta learner. The meta learner is trained on all examples of the data. He is trained on the class probability predictions of the base learners, where he compares the predictions to the correct labels for each example, to discover the strengths and weaknesses of each of the base learners and how to best combine their predictions. In some of our stacking approaches, the meta learner will additionally use the original input features as meta-features, in order to learn and find a good combination between the base learners’ class probability predictions and the original input features for a final prediction.

Once the meta learner for the stacking approach has been trained to learn the best combination of the base learners, the base learners are re-trained on the entire data used for training the stacking approach. That is done for improved accuracy of the base learners - and therefore improved performance of the stacking approach.

4.3 Evaluation

For estimating the performance of the individual classifiers and different stacking approaches on the evaluation datasets, we use classification error rates. That is simply the fraction of labels that are incorrectly predicted, using a validation set or a test set.

When comparing and evaluating the improvement of using one classifier / stacking approach $L_1$ versus using another classifier / stacking approach $L_2$, we calculate the relative improvement in accuracy as shown in Equation 4.1. This same approach has been used in similar studies, such as the one conducted by Džeroski and Ženko [14].

$$ improvement = 1 - \frac{\text{classification_error_rate}(L_1)}{\text{classification_error_rate}(L_2)} $$

(4.1)

4.3.1 Evaluation of the individual classifiers

For each of the five classifiers mentioned in Section 4.2.1, we evaluate their performance individually. For each of the ten evaluation datasets, the performance of the individual classifiers are evaluated using regular 10-fold CV. The performance
of a classifier over a single dataset is measured as the average classification error rate over the 10 folds. We do that to find the best individual performing classifiers for each dataset. That way, we can compare the results of selecting and using the best individual classifier on a dataset, to using a stacking approach on that same dataset.

In order to make sure all methods are trained on the exact same folds, and to be able to reproduce the results from the experiments, a pre-defined seed is used for the random state of the classifiers and the CV splits.

### 4.3.2 Evaluation of the stacking approaches

For evaluating the different stacking approaches we use the same approach as when evaluating the individual classifiers. For each evaluation dataset, the performance of the stacking approaches are evaluated using regular 10-fold CV. It means that each evaluation dataset is split into ten parts, where nine parts are used as a training set and one part is held out as a validation set. The stacking approaches are built and trained on the training set according to the procedure described in Section 4.2.4. The validation set is then used to assess the classification error rate of the stacking approaches. This is repeated ten times, or until all parts of the dataset have served as a validation set. The performances of the stacking approaches over a single dataset are finally measured as the average classification error rate over the 10 folds.

As with the evaluation of the individual classifiers, the same pre-defined seed is used for the random state of all the classifiers used for the stacking approaches, as well as for the random rate of the CV splits.

### 4.4 Statistical analysis

To evaluate the significance of the difference in performance between the classifiers and stacking approaches we perform statistical significance tests. When comparing more than two classifiers over multiple datasets, it is recommended to use the Friedman test to rank the classifiers and to measure if there is a significant difference between them. If a statistical significant difference is detected between the groups (classifiers), a corresponding post-hoc test can be done to find between which groups there is a significant difference [41].

In our analysis, we begin by finding the best base-level learners, with the lowest average classification error-rate over the 10-fold CV, for each dataset. We then start with the null hypothesis that there is no statistical difference between the error rates of the best base-level learners and the stacking approaches. We perform the Friedman test to calculate the rank for the methods on each dataset, where the best performing method gets the rank 1 and worst performing method gets the rank 5 (as we are comparing five methods: best base-level learners and four stacking approaches). The average rank for each method, over all datasets, is then computed and presented. From the average ranks, the significance between the groups can be
computed. In our case, we use a significance value \( \alpha = 0.05 \), as is commonly used in comparative studies \cite{14}\cite{15}. If the p-value from the Friedman test is lower than \( \alpha \), we reject the null hypothesis. If there is a significant difference between the groups, we proceed with Holm’s post-hoc test. Using Holm’s procedure we can measure if there is a significant difference between each of the methods \cite{41}.
Chapter 5

Results

This chapter presents the results from our experiments. In Section 5.1 the classification error rates for all the base learners and stacking approaches, over all ten datasets, are given. In Section 5.2 we compare the performance of the different stacking approaches to the best base-level learners for each dataset. In Section 5.3 we look at the performance of the SGBM2 versus the other stacking approaches. Finally, in Section 5.4 we look at the significance of the results from our experiments.

5.1 Overall results

The average classification error rates of the different individual base-level learners and stacking approaches (the different meta-level learners and their different input meta-features), for all ten datasets, are presented in Table 5.1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Base-level learners</th>
<th>Stacking approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB</td>
<td>LR</td>
</tr>
<tr>
<td>AC - Album</td>
<td>15.89</td>
<td>3.87</td>
</tr>
<tr>
<td>AC - Track</td>
<td>24.15</td>
<td>3.58</td>
</tr>
<tr>
<td>Adult</td>
<td>20.51</td>
<td>20.47</td>
</tr>
<tr>
<td>Breast-W</td>
<td>4.29</td>
<td>4.29</td>
</tr>
<tr>
<td>Heart-C</td>
<td>18.83</td>
<td>18.20</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>11.40</td>
<td>11.69</td>
</tr>
<tr>
<td>MAGIC</td>
<td>27.33</td>
<td>21.03</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>28.60</td>
<td>30.38</td>
</tr>
<tr>
<td>Vote</td>
<td>6.00</td>
<td>3.47</td>
</tr>
</tbody>
</table>

39
5.1.1 Details on the artist chooser results

In Figures 5.1 and 5.2 we show the classification error rates for each of the ten folds of the CV, for the best base learner (Random Forest in both cases) and the different stacking approaches, on the artist chooser’s album and track datasets, respectively. We can see that SGBM2 has the lowest error rate on all folds, for both datasets.

**Figure 5.1.** Err. rates for RF and stacking appr. for each fold on the album dataset.

**Figure 5.2.** Err. rates for RF and stacking appr. for each fold on the track dataset.
5.2 Stacking versus best base-level learners

From the results of Table 5.1 we can see that in most cases, the stacking approaches have an overall lower classification error rate, averaged over the 10-fold CV, compared to the individual classifiers. To get a better picture of this, we do a pairwise comparison between all the stacking approaches and the best base-level learner, for each dataset. Table 5.2 presents the results of those comparisons. For each dataset, we find the best base-level learner (the model with the lowest average classification error rate), and for every stacking approach, we calculate the relative improvement in accuracy of using that approach compared to using the base-level learner.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SLR1 v. BBL</th>
<th>SLR2 v. BBL</th>
<th>SGBM1 v. BBL</th>
<th>SGBM2 v. BBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC - Album</td>
<td>+3.19</td>
<td>-25.94</td>
<td>+5.09</td>
<td>+7.52</td>
</tr>
<tr>
<td>AC - Track</td>
<td>+1.10</td>
<td>-39.70</td>
<td>+2.58</td>
<td>+4.91</td>
</tr>
<tr>
<td>Adult</td>
<td>+0.95</td>
<td>-52.20</td>
<td>+0.81</td>
<td>+3.73</td>
</tr>
<tr>
<td>Australian</td>
<td>+5.32</td>
<td>+2.13</td>
<td>-13.83</td>
<td>-7.45</td>
</tr>
<tr>
<td>Breast-W</td>
<td>-3.70</td>
<td>0.00</td>
<td>-11.10</td>
<td>+25.90</td>
</tr>
<tr>
<td>Heart-C</td>
<td>+3.72</td>
<td>+1.71</td>
<td>-12.52</td>
<td>-7.15</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>+20.93</td>
<td>0.00</td>
<td>+12.44</td>
<td>+20.81</td>
</tr>
<tr>
<td>MAGIC</td>
<td>+3.01</td>
<td>+3.83</td>
<td>+3.75</td>
<td>+5.80</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>+23.68</td>
<td>+27.29</td>
<td>+50.91</td>
<td>+50.91</td>
</tr>
<tr>
<td>Vote</td>
<td>-6.24</td>
<td>0.00</td>
<td>-39.88</td>
<td>-26.33</td>
</tr>
</tbody>
</table>

From these results we can observe that all the stacking approaches perform in majority of the cases better than or equal to the best base-level learners. It is also interesting to note from these results that SLR1 - stacking with logistic regression using the base learners’ class probabilities as meta-features, performs in most cases better than SLR2 - stacking with logistic regression using in addition to the base learners’ class probabilities, the original input features as meta-features. SGBM2 - stacking with gradient boosting machine using the base learners’ class probabilities and the original input features as meta-features, performs however in all cases better than or equal to SGBM1 - stacking with gradient boosting machine using only the base learners’ class probabilities as meta-features. These results may indicate that: 1) when using a linear meta-level learner, additionally using the original input features with the base learners’ class probability predictions as meta-features can often simply confuse the meta learner, lower the importance of the base learners’ predictions and decrease the overall performance of the stacking approach; 2) when using a non-linear meta-level learner, combining the original input features with the base learners’ class probability predictions for the meta learner’s input can help the meta learner and improve the overall performance of the stacking approach.
If we focus on the artist chooser datasets we observe few interesting things. The baseline solution for the artist chooser datasets, Random Forest, is the best base-level learner on both datasets. We observe, as was shown in Section 5.1.1, that stacking with a non-linear meta-level learner (GBM) performs better than stacking with a linear meta-level learner (LR) on the two datasets, regardless of the input meta-features type. Furthermore, we observe that SGBM2 - stacking with gradient boosting machine, using the base learners’ class probabilities and the original input features as meta-features, has the highest relative improvement on the two datasets compared to the baseline solution, making it the best stacking approach, among the evaluated approaches, for the artist chooser problem. To evaluated how well our results from the artist chooser problem generalize, we compare the best approach, SGBM2, to the other stacking approaches, on all datasets. We look at the relative improvement in accuracy of using SGBM2 compared to the other stacking approaches. The results are presented in the next section.

5.3 SGBM2 versus the other stacking approaches

In Table 5.3 we look at how SGBM2 compares to the other stacking approaches. We observe that SGBM2 performs in majority of the cases better than any of the other stacking approaches. SGBM2 performs nine times better than SGBM1, seven times better than SLR2 and six times better than SLR1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SGBM2 v. SLR1</th>
<th>SGBM2 v. SLR2</th>
<th>SGBM2 v. SGBM1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC - Album</td>
<td>+4.47</td>
<td>+26.57</td>
<td>+2.56</td>
</tr>
<tr>
<td>AC - Track</td>
<td>+3.85</td>
<td>+31.93</td>
<td>+2.39</td>
</tr>
<tr>
<td>Adult</td>
<td>+2.80</td>
<td>+36.75</td>
<td>+2.94</td>
</tr>
<tr>
<td>Australian</td>
<td>-13.48</td>
<td>-9.78</td>
<td>+5.61</td>
</tr>
<tr>
<td>Breast-W</td>
<td>+28.54</td>
<td>+25.90</td>
<td>+33.30</td>
</tr>
<tr>
<td>Heart-C</td>
<td>-11.29</td>
<td>-9.01</td>
<td>+4.78</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>-0.15</td>
<td>+20.81</td>
<td>+9.56</td>
</tr>
<tr>
<td>MAGIC</td>
<td>+2.89</td>
<td>+2.05</td>
<td>+2.14</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>+35.68</td>
<td>+32.48</td>
<td>0.00</td>
</tr>
<tr>
<td>Vote</td>
<td>-18.91</td>
<td>-26.33</td>
<td>+9.68</td>
</tr>
</tbody>
</table>

5.4 Statistical analysis of the results

To evaluate the significance of the results from our experiments we perform statistical analysis, as was described in Section 4.4. We use the Friedman test to rank the performance of the best base-level learners and stacking approaches over all
datasets. If the \( p \)-value from the test is lower than 0.05, there is a significant difference between the approaches. Table 5.4 presents the results from the Friedman test, where the average ranking for the approaches are shown. Rank 1 is given to the best approach on a dataset, the second best approach gets a rank 2 and so on.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGBM2</td>
<td>2.05</td>
</tr>
<tr>
<td>SLR1</td>
<td>2.6</td>
</tr>
<tr>
<td>SLR2</td>
<td>3.25</td>
</tr>
<tr>
<td>SGBM1</td>
<td>3.45</td>
</tr>
<tr>
<td>BBL</td>
<td>3.65</td>
</tr>
</tbody>
</table>

The \( p \)-value computed by the Friedman test is: 0.1359. This means that we cannot reject the null hypothesis, that states there is no statistical difference between the classification error rates of the best base-level learners and stacking approaches. What we can do is observe the results from the test. We can see that all the stacking approaches ranked higher than the best base-level learners. We can also see that SGBM2 ranks the highest among the stacking approaches. Furthermore, as discussed in previous sections, we can see that SGBM2 ranks higher than SGBM1, and SLR1 ranks higher than SLR2. These observed results might indicate that there is a true difference between the approaches. They might indicate that stacking with a non-linear meta-level learner, using the base learners’ class probabilities and original input features as meta-features, is in fact the best stacking approach among the evaluated approaches. But we still cannot exclude that the observed difference might be due to a chance. Therefore, our results should simply be taken as such, as possible indications of how different stacking approaches, using linear and non-linear meta-level learners with different types of meta-features, might perform on similar prediction problems, and what approaches might work the best.
Chapter 6

Conclusions

In this work we have evaluated five different classifiers and four different stacking approaches on the two artist chooser datasets, as well as on eight datasets from the UCI Machine Learning Repository. We observe from the results of our experiments that all the stacking approaches perform better than the best base-level learners, despite the fact that comparing specific stacking approaches to the best base learners, for each dataset, gives a bias against the stacking approaches (due to the number of base learners). Among the evaluated stacking approaches SGBM2 - stacking with gradient boosting machine using a combination of the base learners’ class probabilities and the original input features as meta-features, performed the best. The next best performing approach was SLR1, stacking with logistic regression using only the base learners’ class probabilities as meta-features. It is interesting to note from these results that using the original input features as additional meta-features increases the performance of the stacking approach when the meta learner is non-linear, while often decreasing the performance when the meta learner is linear.

Although we cannot reject the null hypothesis that there is no statistical difference between the classification error rates of the best base-level learners and stacking approaches, according to the results from the Friedman test, we can observe the results and use them as possible indications of how the different stacking approaches might perform on similar binary classification problems, and what approaches are likely to perform the best. Overall, the results from our experiments indicate that stacking with a non-linear meta-level learner, using a combination of the base learners’ class probabilities and the original input features as meta-features, performs better than stacking with a linear meta-level learner, regardless of the type of meta-features used for that learner. These indications further support Džeroski and Ženko’s results [14], who concluded that using multi-response model trees as a meta learner for stacked generalization performs better than using multi-response linear regression as a meta learner.

On a final concluding note, we think it is important to quote Wolpert and Macready’s no-free lunch theorem [28] stating that no single approach can be better than another for all classification tasks, and the same thing goes for stacking
CHAPTER 6. CONCLUSIONS

approaches. It is always a good strategy to test different methods on the problem at hand, as all classification tasks are different. In this thesis, we have evaluated four stacking approaches, over ten different datasets. We have shown that using a non-linear classifier, gradient boosting machine, can be a good choice when selecting a meta-level learner for a stacking solution on a binary classification problem, especially when the original input features are combined with the base learners' class probabilities as meta-features. This way, the meta-level learner is able to capture non-linear decision boundaries of the features and the strengths of the different base learners in different sections of the input space.

For the rest of this chapter we have a detailed discussion in Section 6.1 on the advantages and disadvantages of stacking, as well as on some of our findings on the artist chooser problem. We finally suggest future work in Section 6.2.

6.1 Discussion

In this section we discuss in more detail some of the findings from our experiments and when stacking can be a good choice of approach for a prediction problem.

6.1.1 The computational complexity of stacking

We have shown, as other studies have done, that using stacking can improve performance over individual classifiers in many cases. Using a stacking approach over a single classifier comes however with its costs, particularly: computational complexity. If we assume, like in our experiments, that we use 10-fold CV for our stacking approach and five base learners, then we need to train each base learner ten times on 90% of the training data and one time on the whole training data. Additionally, we need to train the meta learner once. This means that the training time for the stacking approach would take roughly 50 times longer than the training time for a single classifier. Training a stacking approach with cross-validation can however be done in parallel, as each fit on a fold is independent. That could reduce the training time in this case ten fold.

When considering the training time of a non-linear meta learner compared to a linear meta learner, it was a bit longer in our experiments. Using the original input features as meta-features increased the training time of the meta learners a bit as well. These increases in training time were however only a small fraction of the overall training time for the stacking approaches, as we trained all folds sequentially. Therefore, these increases did not affect our training time much, but it is still good to be aware of these differences.

Regarding the prediction time of a stacking approach versus a single classifier, we did some comparisons using a trained SGBM2 stacking approach and a trained Random Forest model on the artist chooser album dataset. The average prediction time on an album example using the SGBM2 approach was roughly twice as long as for the RF model: 1.1502 ms. compared to 0.5822 ms. This difference is only
6.2. FUTURE WORK

a very small part of the total prediction time for the artist chooser solution and would unlikely matter, but could matter in other classification tasks.

Although the stacking approaches in our experiments showed in many cases good relative improvement in accuracy over the individual base learners, the increased improvement in percentage points may not always be worth the increased complexity of the approach. Stacking is a good approach for pushing the performance limits on prediction problems, but the improvements of the approach versus the cost needs to be independently evaluated for each case.

6.1.2 The artist chooser problem

In this work we have evaluated five different classifiers and four different stacking approaches on the artist chooser datasets. We report that among the base learners, a Random Forest model performed the best. Among the stacking approaches, SGBM2 performed the best - and better than the best base learner. We explored the properties of the datasets, looked at feature correlations and feature importances. We learned that some features, which we assumed would be important for the model, had very low scores and were not important, for example the track language feature, and vice versa; some features, which we assumed would not be that important, were in fact important.

We also did some analysis on the performance of one of the individual base learners (RF) to inspect the rate of false-negative predictions. False-negative predictions mean that the model decides not to assign an album to a candidate artist - when it actually should. This results in a new, duplicate, artist being created in the system, which is harder to fix than it is to move an album from an incorrect artist to the correct artist. By lowering the threshold of the model, it classifies more instances of tracks/albums to the 'assign' class, and therefore minimizes the false-negative predictions.

6.2 Future work

The work in this thesis can be extended in different ways. For one, more datasets can be added to further evaluate if there is a true difference between the stacking approaches. More stacking approaches can be evaluated as well to see if there are other non-linear meta learners that perform better than gradient boosting machine.

For the artist chooser problem it would be interesting to make a process that gathers together all the difficult cases that the model fails to correctly predict in production and add them to the list of edge cases included in the training data pipeline. Although the validation scores of the artist chooser models are high, there are always difficult cases that are hard to predict. Trying to include more of those edge cases and less-popular artist examples at training time could help further improve the overall quality of the artist content matching.
Appendix A

Software Packages

The following software and Python packages were used in this study: IntelliJ IDEA (2017.3.3), Python (2.7), avro (1.8.2), matplotlib (2.2.2), numpy (1.14.0), pandas (0.22.0), pmlb (0.3), scikit-learn (0.19.1), scipy (1.0.0) and seaborn (0.8.1).
Bibliography


Declaration

I hereby certify that I have written this thesis independently and have only used the specified sources and resources indicated in the bibliography.

Stockholm, June 27, 2018

Fannar Magnússon