# PREDICTION OF PROJECT SPENDING FOR AN NGO

Industrial Engineering and Management Thesis of
Matthijs Nijholt

Supervised by:

Dr. Ir. W.J.A. van Heeswijk
Dr. A. Abhishta

## Management Summary

The research of this thesis is done for an anonymous worldwide Non-Governmental Organization with an office in the Netherlands (NGO). The goal of this thesis is to predict the final spending and the deviation of the prediction of their projects, on an activity level, based on plans. The NGO is interested in this as they are implementing a system called LuKE. This is done so donations given for restricted set of projects can be matched to one or more of those projects during execution instead of after the projects are done. To do this well the NGO is interested in knowing how much money they can confidently assign to a project. In order to find this amount we look into predicting the project spending, to find out if Machine Learning (ML) models are able to better predict spending than manual budget. We also look to see if any new insights can be gained into the spending on the projects through our research. At this moment in time with limited resources and insufficient data available this is not yet possible, although we do think this can be possible in the future and give advice to get to this point.

Several departments are influenced by the predictions. Finance is responsible for managing cashflows, auditing and ensuring dedicated funds are spent on the correct projects. Field executes the projects and can be considered the operations department of the NGO. Development is the department which raises money, with management overseeing the entire NGO. While the main benefactor of better predictions would be Development which raises money, other departments could also benefit from better predictions. Management would benefit from having more certainty when approving plans, Finance would be able to improve cashflow plans and Field would have an easier time overseeing their projects. Everybody would benefit from an easier end-of-year reporting, as planned and actual spending would be more aligned. For this purpose three types of predictions would be useful: 1. Predicting the final spending before the start of the project, 2. Predicting final spending during the execution of the project, 3. Predicting the spending pattern of the project before the start of the project. As the available data does not allow for the second type of prediction, we attempt to predict the final spending before the start of the project and predict this for each month the project runs for.

Our approach is to use Machine Learning (ML) to tackle this problem. ML is able to find both basic patterns as well as complex patterns experts are not able to find. The capabilities of ML line up with the demands of the predictions, which is why this group of models and algorithms is used. We looked into models others had used in similar situations and found that models such as K Nearest Neighbors and Neural Networks are commonly used. The way to combat Machine Learning problems like overfitting and unequal variances in the errors for these models was also looked into. The model which had the most overfitting, which is when a model becomes too focused on training data and loses the ability to predict new data, was the Neural Network. This was far worse than other models, but this was considerably less so when using the Sparse or Dropout variants. These models could be expanded to be able to find more and more complex patterns, but this took too many resources to train. The other problem of unequal variances of the prediction errors could be reduced or even removed by introducing a scaled prediction goal, for example instead of predicting an amount of dollars spent, a percentage of the budget spent would be the prediction. However these different goals meant that the average amount of dollars a prediction was off by, over 6000 USD, was several times the difference between the budget and the actual spending roughly 2000

USD, which is what our model is supposed to be better than. Applying these methods led to models which required too much processing time or reduced the accuracy so that the predictions became much worse than the budget accuracy.

From our experiments on the aforementioned models as well as other models, it became clear that monthly or quarterly predictions were not accurate enough, as the Mean Absolute Error was 85% of the money spent in the best case, and usually over 100%. This means that on average a prediction was wrong by 85% of the actual value, either too much too little. Quarterly predictions summed up and used as yearly predictions were almost as accurate as yearly predictions. Predicting this way is less prone to overfitting and has the potential to be useful in the future. Therefore we choose to have our models predict yearly spending by training them to predict quarterly spending. We introduce a genetic algorithm adjusted from literature to find good parameters for models. This was then adjusted so the value of two parameters combined cannot exceed a user determined size. This was done to constrain the computation time while allowing the Genetic Algorithm more possible parameter configurations. We looked at the dataset we had to see if the data was of sufficient quality and properly correlated to the rest of the dataset and found no major problems during this analysis. However prior to this we did find one feature we had to remove as it negatively impacted the prediction accuracy. From the fact that we could only find one bad feature and the inability of our models to beat the budget by any significant margin we conclude that there must be at least one unknown factor which has a big influencing on spending which was not included in the dataset.

From our final optimized models the K Nearest Neighbors model was most successful, which is a simple model looking at the most similar datapoints to the new datapoint in order make a prediction. Small experiments with computationally intensive Neural Networks showed promise that they may beat the K Nearest Neighbors model but not to such an extent that they might be significantly better than original budgets. Given the introduction of LuKE and other planned changes in the way projects are to be planned and executed, we advise the NGO to improve data gathering to include more factors relating to the project spending and wait until a relatively stable workflow with regards to the projects is achieved. After data is gathered for this working situation and possibly with additional computational resources they can revisit the models we prepared for them. In the meantime or as an alternative we propose the solution of requesting periodic updates from experts on the projects which are constraining fundraising, limiting the paperwork as much as possible to reduce the burden. If there are people available we would also recommend asking experts about what factors they think influences the total spending of projects and how to measure those factors.

## Word of thanks.

I would like to thank some people for their direct and indirect contributions to the thesis. Of course my supervisors at the University of Twente have been instrumental in guiding the whole process. I am very thankful for the time and patience from both Wouter and Abhistha when giving feedback and answering questions. I feel very fortunate to have had these supervisors which helped me throughout me thesis and who I could always bounce ideas off. I also had supervisors from the NGO, both of whom I also apricate greatly. They helped me find my way within the NGO and see how my research fit within the entire organization. I am thankful for their effort as well as mental checkups and kindness.

I would also like to thank my roommates I had in the office. I enjoyed our small talk, more serious discussions and jokes a lot and missed you all when we had to go into corona lockdown. Besides my roommates I also missed the rest of the colleagues, as everyone in the office was kind and I had great talks with so many of you.

I would also like to thank my friends from NSE whom cheered me up when I was having a bad week, encouraged me when I needed it and celebrated when I had some good news to share. Thanks for all the fun we had and for keeping me sane.

Finally, I would like to thank my parents and sister for their support and interest in my thesis, even when really had no clue what I was talking about. They listened and gave advice when needed and helped me balance life during lockdown.

# Table of Contents

# 1. Introduction

This chapter will introduce the problem experienced by a Non-Governmental Organization (NGO), which will remained unnamed throughout this thesis. For privacy and security reasons it will be referred to as the NGO. In addition, all US dollar amounts in regard to the NGO have been scaled by a constant which can be found in Appendix A, which will not be publicly available. There are exceptions to this which will be clearly marked when this is the case. Chapter 1 starts with an introduction to the relevant parts of the way the NGO operates to understand the need for a predictive model. This is followed by the analysis of the core problem and an explanation how the predictive model will improve on this problem. The last part of this chapter contains the research goal, deliverable and performance indicators.

## 1.1 Background

The NGO is an organization operating worldwide with an office in The Netherlands. The focus of this NGO is helping certain marginalized peoples. They have different kinds of projects such as food parcels, literacy trainings, trauma counseling and literature distribution. For the purpose of this research there are five important departments that need to be considered. They are shortly introduced here and explained further in Chapter 2.1. The first is Management, which consist of the board and highest level of managers in the organization. The next department is Finance, which among the regular tasks needs to ensure that spending documentation is done correctly so governments and donors can know that the money has been spend well. Executing projects can be considered the Operations of the NGO, however within the NGO this is usually referred to as the Field. One of the other core parts of the organization is fundraising. The combined task of fundraising, raising awareness and keeping the donators informed is referred to as Development. They share an important connection with the Field as the Field is reliant on Development to raise the money the Field wants to spend on their projects. One of the struggles for the Finance department is that roughly 50% of the money that Development raises is given for a dedicated project, or a specific range of projects. Development is raising money for the projects during the execution of those projects. If these projects end up underspending there may be unspent money leftover meant for specific projects. Making sure that all the specific funds are allocated to projects that meet the requirements of the funds used to be a very time consuming and tough job for the Finance department. Project information organizes all the information regarding the different projects and activities that are going on. They are the central hub when any of the other departments need information regarding projects.

The NGO is introducing a new software system to guide how Management, Finance, Field and Development work together, which will be referred to as LuKE in this thesis. In previous years the fundraising was done in a way where Development was a main driver, with the fundraisers getting some Field projects proposals assigned from Management but also requesting more Field project proposals when they saw fundraising opportunities. Often Development requested more projects than they could raise funds for to give themselves some flexibility when talking to donors. Development was mostly focused on raising as much money as possible without paying enough attention whether the Field would be able to spend it, or whether they could find funding for all the projects. This was bad for the morale of the Field as they spend considerable amounts of time

preparing project proposals that ended up unfunded. Other projects were overfunded which caused problems for Finance, Field and Management as projects needed to be scaled up, or dedicated funds needed to be moved around while there might be more important projects to put time and effort in.

The new LuKE system is driven from the Management and Field and is part of a bigger change to focus on the goal to serve the right needs and to play to Fields capabilities and strengths. Development will be assigned or can promise to raise money for projects that Field has proposed and that have been approved by the relevant management. The aim of the new way of working is two-fold. The first major goal is to ensure that money raised by Development can easily be allocated to the projects they fit with. The main way this is done is by raising money that is as unrestricted as possible. An example of this would be that where previously money would be raised for food parcels for Syria, Development will now try to reduce the specificity by raising money only restricted by the either the country or project type, so either for Syria or for emergency relief. Matching the funds raised by Development to projects is still the responsibility of Finance, however with the new way of raising funds pressure of matching grants to projects is expected to be significantly less.

The second goal is allowing the organization to better focus on the projects in line with strategic priorities. In many of the countries the NGO is operating in there are various sources of instability and a complex nature of the needs. This makes determining the best mix of projects complex. Giving direction to the Field projects becomes easier for Field directors and global directors when Field is leading in fundraising. They can request the offices to work on a project mix completely in line with the long-term strategy that has been chosen. The tradeoff is that this might come at reduced performance of Development, who may find it harder to raise money for the projects they get assigned. This could be offset by better performance and reporting of Field if the way the money is spend is better than before. These changes are made easier, or made possible by the introduction of LuKE, as this gives the control to Management and Field to determine projects and shows the amount of dedicated funds a Development team is allowed to raise for a specific project.

## 1.2 Core problem

This new system reduces or completely removes the uncertainty from the Development side for matching donations and grants with projects. The system is reliant on the Field spending being predictable. If Development raises the money which is budgeted but it turns out Field underspends this creates the same kinds of problems for matching grants as the previous system. Ideally when there is the expectation that at least 95% of the budget will be spend Development can safely raise dedicated funds for 95% of the budgeted project costs. There is more control for Management in the new system as they can better decide which projects will be done, rather than Development looking at what projects draw in most donations. The new system also increases the ability to learn over time how much projects are spending, but there is already historical data available on spending, and as such the preference is to hit ground running by utilizing this available data. The main intended benefit from a predictive model is to find how much of the budget of Field can be allocated for dedicated donations. Therefore predicting the actual spending at a low standard deviation is the goal of this thesis.

To solve this problem Machine Learning will be used. There are two big reasons for this. The first reason is that the goal is to find patterns regarding which types of projects end up spending closer to or further from the planned budget. Machine Learning is able to find very complex patterns which people cannot find, while also finding less complex patterns. The NGO wants to learn as much as possible from the data, if possible more than expert knowledge could bring which Machine Learning may do. Machine Learning looks for any pattern it can find, and in this way is able to find patterns which people may not consider. The second reason for using Machine Learning is due to the data structure. Most data are in categories, such as the country and project type. Other forecasting and predictive methods such as linear regression, as the name implies, cannot find non-linear relations. Like Machine Learning, polynomial regression can theoretically approach a function to arbitrarily close, although this does require sufficient training data for both methods. Polynomial regression however, can easily overfit from one or a few outliers, and given the size of the dataset available and the low counts of training samples in select categories we expect a poor performance from this method. Overfitting is when a model loses its ability to predict new data because it becomes too focused on training data. The projects are also executed per one-year time period, and may change, disappear or start from nothing the next year. Some forecasting models which rely on previous time periods may be used for things such as stock forecasting. Unfortunately these cannot be used for most or all of the projects. The combination of goals and data is one which fits Machine Learning particularly well. Therefore the choice has been made to use Machine Learning models to approach this core problem. The definition of Machine Learning used in this thesis is: "Machine learning is the scientific study of algorithms and statistical models that computer systems use to perform a specific task, by feeding them data and information in the form of observations and real-world interactions and relying on patterns and inference instead of using explicit instructions." How this definition was chosen will be explained in Chapter 2.

## 1.3 Research goal, deliverable and performance indicators

To summarize the problem outlined in section 1.2 the main problem the NGO is facing is that they aim to better match Development fundraising and Field spending. A new system is being put in place to use for Development, but the project spending of the Field side of the equation are less reliable restricting the ability to use the new system of Development. This leads to the main goal of this thesis:

Predicting the final spending and the deviation of the prediction of projects of the NGO, on an activity level, based on plans.

To reach this goal a Machine Learning model will be developed that given some input data, such as planned budget and project type, can predict how much money will have been spend on the project when it is completed. This model will be the deliverable of the master thesis project. The model is chosen through comparison of different predictive models to find a model that is most suitable for this case. More details on the deliverable will be given in the dedicated deliverable section in chapter 2. The research questions and sub-questions relating to the research goal are the following:

1. What kind of predictions does the NGO need?
    - Which departments benefit from predictions?
    - What specific type of prediction would be most useful?

2. What models are likely to work?
    - What models have been used for similar problems?
    - How can common Machine Learning problems be prevented?
3. How can we use the models from literature to predict spending?
    - How can appropriate parameters be found?
    - What can be predicted with the models and data available?

Each of these research questions will have their own chapter in chapters 2-4. We will judge our final models on three performance indicators:

1. The accuracy of the model.

2. The standard deviation of the prediction error.

3. Ease of use and future potential

The reasons for the choice of these performance indicators are the following:

Accurately predicting the spending is the main goal of the model, therefore this is the first performance indicator that is considered. However, if the model is almost spot on 80% of the time but completely off the other 20% off time it is normally less useful than a model that is almost spot on 75% the time of the and still come somewhat close the remaining 25% of the time. Therefore, it is important to also consider the standard deviation of the prediction error. Finally, the model is not just a one-time build, it will update with new data, may require some maintenance during its lifecycle. As processing power becomes more available and cheaper over time, models which take too much time to train right now may become feasible. As complex models become more accessible they may eclipse simpler models. As such, when recommending models, we will consider the ease of use for people new to Machine Learning, and the future prospects for the models as well.

When considering our priorities, the first performance indicator is considered to be most important, followed by the second indicator. The third will be mentioned with their pros and cons, although ultimately it is up to the NGO to decide what they feel is more important. If several high performing models are found we will rank them, but if this is not possible we will instead explain when and why our best models are more likely to be useful.

For the first research question, which can be found in Chapter 2, we will be looking at the NGO itself and the past, current and future of the NGO with regards to the accuracy of the budget. We will also be looking briefly at what Machine Learning is as well as introducing the data used for this thesis. The chapter will conclude with a deliverable based on the needs we find from the departments of the NGO.

The second research question will be answered in Chapter 3, where we will be discussing previous research and general literature. We will give explanations to the workings of the different models which may be able to predict project spending. After the general introduction to what Machine Learning is in Chapter 2 we will discuss some techniques, characteristics and problems found in the Machine Learning field. The models will be used as a starting point from which we attempt to

predict spending, while the general Machine Learning theory will be used to explain problems found, how we solved them and why we used that technique. It will also be used to argue which models and what specific type of prediction is most appropriate for the NGO to use in the future.

Our final research question will bridge the gap between the theoretical models and the final models. Adjustments will need to be made to properly use the models, and these will be discussed in Chapter 4. After this chapter we have the models and methods to fulfill the main goal of the thesis: Predicting the final spending and the deviation of the prediction of projects of the NGO, on an activity level, based on plans.

After we answer our research questions we will discuss any interesting analysis results we have as well as our final results in Chapter 5. The conclusion and advice we have for the NGO are discussed in Chapter 6. We will also discuss limitations with our research and any recommendation for further research here.

# 2. Research motivation

This chapter further introduces the NGO by looking at the different departments and how the NGO has been changing in the past years. After this the reasons why the NGO is looking for a better understanding of their spending patterns are discussed to show why this thesis is relevant for the NGO. This chapter ends with the deliverable of this master thesis project and the starting point of the data analysis.

## 2.1 Departments of the NGO

There are several departments that are affected by the accuracy and general spending of the Field department. This subchapter will discuss the relevant departments and how they stand in relation to each other with regards to the main research goal of predicting Field spending. We will first discuss the different departments to explain the main tasks of these departments. After this explanation Figure 1 shows the relations between the between different departments of the NGO, although it is simplified somewhat as the complete relationships between the departments are quite complex.

- Field itself does projects in many countries and spends most of the money that the NGO receives. They make project proposals that need to be approved by the general management and board. When these projects are expected to spend less than budgeted usually there is enough capacity available to spend this money on other projects. If projects look like they will end up costing more than the budget other projects within the region of the overspending project reduce spending and/or more money is sought from donors.

- Development raises money for Field projects. This is done with a mix of specified campaigns, government grants, specified gifting and where-most-needed gifts. This last type is completely free of any restrictions when spending, the others need to be spent on certain projects. With the switch to LuKE there is also more emphasis placed on raising money as unrestricted as possible.

- Management tries to ensure that all offices of all departments function efficiently but also in line with the core values and long-term strategy of the NGO. They approve budget and decide on the long-term strategy. The management answers to the board.

- Project Information handles the information concerning the Field projects. Before LuKE they attempted to keep up-to-date data on all projects and project-proposals and were queried by other departments for information. Especially during the months where next-years planning was made this led to mentally taxing and high-pressure times, but this has been improved greatly with LuKE. They are still responsible for the information management and does things such as bundling project information in convenient packages for donors to help Development.

- Finance needs to ensure that money given for a specific purpose is also spend on that purpose, aside from normal Finance duties. Keeping track of dedicated funds becomes a complicated task when the level of specificity varies, e.g. Syria versus Middle East, and also in type, e.g. literature or emergency relief.

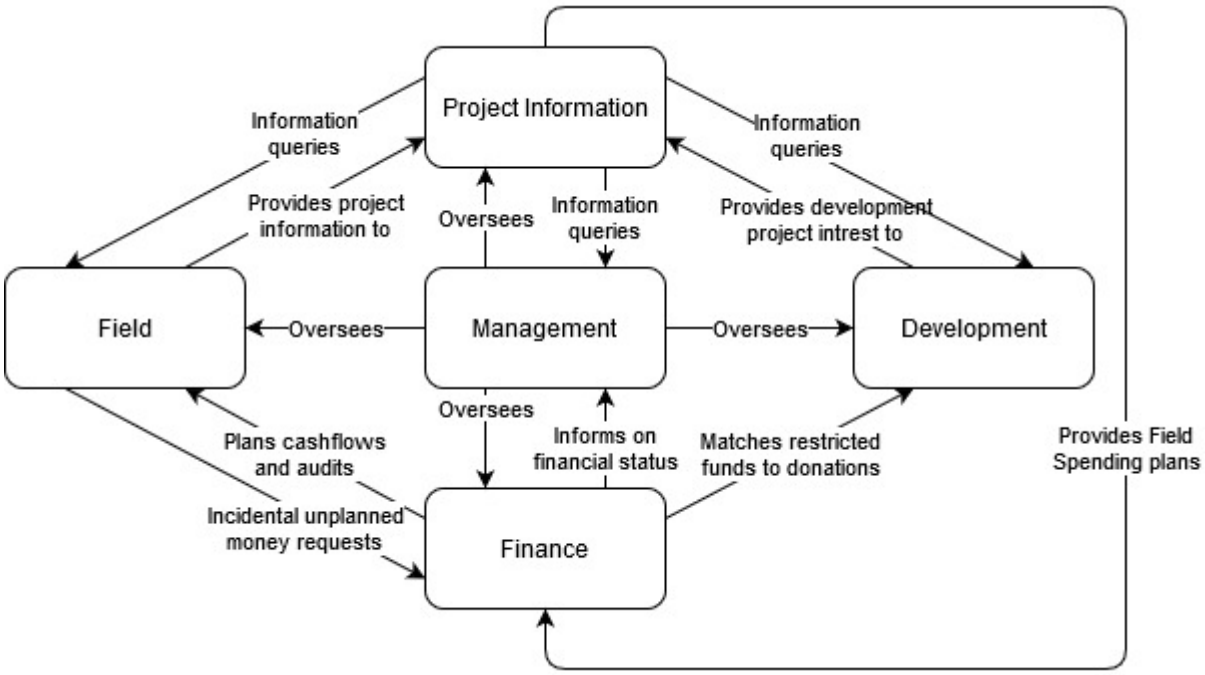


*Figure 1 - Simplified relations between different departments of the NGO*

## 2.2 Old situation

In the previous system of matching donations with field projects there was a huge strain on two departments, the Finance and Project Information departments. The project information department had a large and unbalanced workload because of the large amount of data gathering and checking that needed to be done at the second half of the year to prepare the plans for the coming year. Information regarding how much money had been raised for a certain project was not centrally available, instead local offices each had their own targets without there being a live total amount raised. When making Field budget estimations this was done by expert opinion and previous budgets while adjusting a bit for inflation.

## 2.3 Current situation

In past years it seems that the different departments worked in a more decentralized manner. Priority was given to Field and Development as the most important departments, with mainly Project Information and Finance left to clean up. This is in the process of changing. Through the last years and continuing still, a number of bigger and smaller changes and reorganizations is greatly improving how the different departments are working together. This is done through the introduction of LuKE with, as of the date of this chapter, partial integration with the financial information system. Through the way the departments are required to use this system more control is given to the Management team. With LuKE there is integration with the project database and allows for real-time updates regarding the amount of money that Development will raise. It keeps track of a large amount of information, reducing stress on project information considerably. Since

there is one user-friendly interface one of the problems that occurred previously has reduced considerably, which is that people where using outdated spreadsheets or generally not having proper version control. Nothing has changed regarding the way Field is making their estimations.

## 2.4 Need for better predictions.

While the main reason for the predictions has been given already, after some interviews with several employees from different departments multiple reasons for better Field spending predictions have been gathered.

- The first reason has already been mentioned in the introduction, which is that the NGO is a charity organization and receives money from donors. Therefore it is important that money given for a restricted type of project is spend on such a project. If the money cannot be spent on the original cause in the year it was given for the money may either be forwarded to the next year or if this is allowed by the donor the money may be reallocated. If both are not possible the money will be returned to the donor. Having better knowledge of spending at the beginning of the year will decrease the need for these measures. Having updated projections throughout the year would allow for better communications with the donors and adjustments in the projects Development is raising money for.

- The second reason is that approving Field budgets is easier and can be done with more confidence. Choices need to be made between different projects but when the reliability of the projections increase there are less what-if scenarios and easier decision making for Management and the Board.

- Thirdly a better understanding of the spending will improve cash flow planning. Cash flows are important for many industries, this is also the case for charity organizations. Money does not arrive equally throughout the year, December is the month when most donations are received while the summer months are usually slow. In contrast according to the NGO Field tends to spend somewhat regularly throughout the first 11 months and slows down a bit in December. These very different patterns mean that some reserves and planning is required to ensure the solvency of the organization. As the NGO is aiming for a more just-in-time approach to cash flows better predictions will allow for a better implementation of this approach. There is a second part to the cash flows where spending patterns are important, which is the physical availability of resources. While money can be transferred near instantly between western countries, this is not the case for some of the countries the NGO is working in. Furthermore, supplies or other resources may also take time to arrive. Planning for these flows can be improved by better understanding of the spending patterns.

- The fourth reason is quite straightforward: Project managers of Field can better manage their projects and spending if projections are more accurate. If the budgeting is not reliable keeping track of fund allocating becomes very tough. Both better understanding of the spending patterns of projects as well as accurate predictions throughout the year as to the costs of a project will help in this regard.

- Fifthly better projections lead to easier end of year reporting. The closer the actual spending is to the budgeted amounts the fewer explanations are needed and fewer adjustments will have to be made. Development has an easier time reporting to donors what the money has been spend on which gives confidence to donors.

- Finally, the sixth reason why better predictions are needed is because of (geo) political and natural reasons. As the NGO raises money in many different countries and has Field projects in many countries it is important to adhere to the many different rules, regulations and government requirements, for matters such as accounting and permits. Different spending amounts can lead to different rules and if the project is large enough an external expert may be worthwhile. This is becoming more important as rules become tighter and the international relations become more tense. Although natural disasters are not the primary reason for the NGO to start projects in a certain, it is a reason for the NGO to begin work, start new projects, or increase projects in the affected area. As the Centre for Research on the Epidemiology of Disasters has found an annual average increase in disasters of 8.4% between 2000 and 2007, it clearly suggests an increase in natural disasters. The projects done as a result of natural disasters are thought to underspend, with Development being reported of often raising much more than is needed, 2-10 times the requested amount. These projects can benefit from some guidelines as to the appropriate budget and for Development it would be interesting to better understand how much money will be raised for such projects.

## 2.5 Definition of Machine Learning

As mentioned in chapter 1.2 Machine Learning will be used in this thesis to predict Field project spending. The main reasons for Machine Learning are the way it fits the data and goals as well as the ability to find non-linear relationships, as well as patterns humans may miss. The field of Machine Learning (ML) is relatively new, and as such it is not surprising that no consensus has yet been reached on the definition of Machine Learning. In recent highly cited articles few, if any, discuss what ML is, instead focusing on their new contribution to a field and expecting that the definition their reader has is close enough to theirs to not cause confusion. Outside of scientific papers some attempts have been made to define Machine Learning. The following table contains several definitions, one from Wikipedia and one from an Emerj.com article where Emerj CEO Daniel Faggella attempted to make his own definition by collecting and combining the definitions of several reputable sources, all of which are also included in this table.

| Source | Definition |
|---|---|
| Wikipedia (Machine Learning, n.d.) | Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. |
| Nvidia (Michael Copeland, 2016) | Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world. |
| Stanford (Machine Learning, n.d.) | Machine learning is the science of getting computers to act without being explicitly programmed. |

| | |
|---|---|
| McKinsey & Company (Pyle & San José, 2015) | Machine learning is based on algorithms that can learn from data without relying on rules-based programming. |
| Pedro Domingos, University of Washington (Domingos, 2012) | Machine learning algorithms can figure out how to perform important tasks by generalizing from examples. |
| Tom M. Mitchell, Carnegie Mellon University (Mitchell, July 2006) | The field of Machine Learning seeks to answer the question "How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes? |
| Yoshua Bengio, Université de Montréal (Bengio & Faggella, 2019) | Machine learning research is part of research on artificial intelligence, seeking to provide knowledge to computers through data, observations and interacting with the world. That acquired knowledge allows computers to correctly generalize to new settings. |
| Danko Nikolic, CSC and Max-Planck Institute (Nikolic & Faggella, 2018) | Machine learning is the science of getting computers to act without being explicitly programmed, but instead letting them learn a few tricks on their own. |
| Roman Yampolskiy, University of Louisville (Yampolskiy & Faggella, 2019) | Machine Learning is the science of getting computers to learn as well as humans do or better. |
| Daniel Faggella (Faggella, 2019) | Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions. |

Common themes in these definitions are learning, generalizing and doing this without explicit instructions, which are the core of Machine Learning. While the definition that Daniel Faggella (Faggella, 2019) has some good parts, it seems partially incorrect, in particular "Machine Learning is the science of getting computers to learn and act like humans do". While some algorithms mimic human learning, this is only because in those cases it yields the best results, rather than wanting to copy human learning and behavior. There are numerous algorithms which do not attempt to copy human behavior and perform well. And most importantly, the goal of the methods is not to learn as well as humans, but as well as possible, trying to and sometimes succeeding in surpassing humans. The definition of Wikipedia (Machine Learning, n.d.) does not contain any false statements but is not very specific and could use some more explanations as to how ML learns to better explain to non-experts how Machine Learning is able to perform the specific task. This is something that Daniel Faggella does explain nicely, and combining elements from these sources leads to our definition of Machine Learning used by this thesis is as follows:

Machine learning is the scientific study of algorithms and statistical models that computer systems use to learn how to perform a specific task, by feeding them data and information in the form of

observations and real-world interactions and relying on patterns and inference instead of using explicit instructions.

## 2.6 Datasets

As will be further elaborated on in Chapter 3 the current literature on NGOs is quite limited and no directly related studies have been found. Therefore research will need to start at the ground level, and more basic models and relations between variables will have to be researched before appropriate complex models can be selected. Given the way the data is structured and the difference between countries which the NGO works in four different countries have been selected as a smaller dataset for analyses, these are Egypt, Ethiopia, Nigeria and Vietnam. The NGO has sizable projects in these countries which is why these countries are considered. The expectation is that due to the size the impact of minor random events will be among the lowest of the countries. Furthermore, there are some major differences between them. While Egypt, Ethiopia and Nigeria are all in Africa, Egypt differs by having a Middle Eastern culture while Nigeria is a relatively unstable country for the NGO's projects. With the less stable nature of Nigeria some insights may be gained in the impact of larger unpredictable events. Ethiopia is more stable with an African culture allowing for some more fair comparisons with Nigeria, while Vietnam differs from the rest in their Asian culture. Therefore these four countries seem to be suitable candidates to give an indication if one model could be found or if the differences between the countries are to great and several are needed. Chapter 4 will report on the findings of this analysis, as chapter 3 will first discuss relevant literature.

The second dataset contains projects from nearly 60 countries and contains over 4400 datapoints, which in the context our research are 4400 unique projects. For both datasets a project is unique in the combination of type of project and year, as some projects are repeated over several years. The goal will be to make a model which can predict this historical data which hopefully will also be able to predict future spending patterns. This dataset does omit some countries of which the data is not available in sufficient quality. Both datasets do not consider any projects that have been canceled, since the goal is to predict spending, not cancelation, and projects that had a difference of more than 100.000 USD in comparison to the budget have been taken out of the dataset. If any projects are decreased or increased in size by that much it needed to have gone through management and is most likely due to some major event, which means it is extremely hard or impossible to predict. As such big differences can strongly impact Machine Learning models, and often in a bad way, the choice is made to remove the few that met these conditions.

## 2.7 Deliverable

There are three kinds of improved predictions which would help the NGO:

1. Better predictions at the start of the year
2. Better predictions throughout the year
3. An understanding of the spending pattern throughout the year.

The difference between the second and third type are that the second implicitly uses the spending pattern to make predictions while the third type explicitly explains what spending pattern is likely to occur. The focus will be given to the first and third type of predictions: Predicting how much a Field

project will cost before it is started and during execution. This should improve the efficiency throughout Field, Development, Finance and Management and should also improve the confidence of donors. However, reporting throughout the year is not done in as much detail as needed for daily or weekly predictions, the number of years of data is limited and sometimes is absent completely. This makes it impossible to make the second type of predictions.

The deliverable is to make a model that can estimate real costs based on the budget and project variables.

The model will take project data of the NGO's standardized format and calculate what it expects the project will actually spend. This will include an estimated standard deviation. While this paragraph is talking about a singular model to predict everything this does not mean that the end model will have just one predictive model, it may happen that different predictive models will be used for different variables, e.g. different per country.

# 3. Literature

This chapter discusses literature related to finding a model with which the spending of Field projects can be predicted. The chapter starts with a short section on literature related to our problem. People already familiar may look at Figure 2, which gives an overview of the topics discussed by using technical terms instead of the real section names. This will allow people who know something about Machine Learning to find interesting topics. We start by explaining what machine learning is in general and go into detail on some of the Machine Learning topics important for this thesis. This will explain topics such as what kind of Machine Learning problem this is. After explaining ML in general we discuss techniques used to make ML models work properly. These are needed either because of the practical limitations of ML or to prevent common/inevitable problems. These concepts and techniques are used to explain models in the next section of this chapter. We end by comparing the different models and explaining why we chose to include them in this thesis

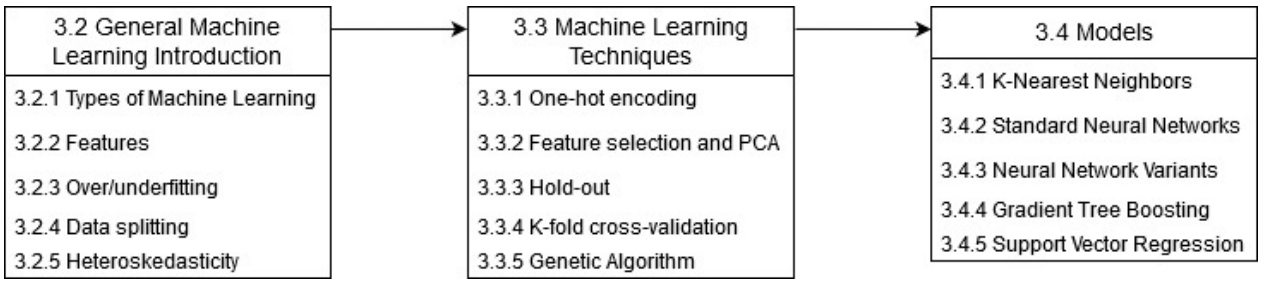


*Figure 2 - Structure Chapter 3.2 - 3.4 with technical section names*

## 3.1 Related literature

Research concerning Non-Governmental Organizations or charities is not as common as industry, at least not regarding spending, but from what has been seen this holds true for most parts of NGOs. We looked at research by using different keyword combinations of charity, NGO, Non-Governmental Organization and Machine Learning, predictions/predicting. The only part that does seems to have been researched somewhat thorough is the psychological part of giving to NGOs or charities. There have been a few researchers that tried to predict the amount of donations charities will receive by means of machine learning, but no studies on predictions of other topics such as spending have found. This does make sense as machine learning is a more recent field and the pressure on NGO's and charities to perform optimally is considerably lower than most publicly traded companies. This means that the research performed here will be quite novel and does not have much prior knowledge to learn from. There is no previous knowledge on common problems or shortcomings with data. This makes it harder to anticipate on problems, and when encountering atypical model behavior makes it harder to pinpoint the exact reason.

When looking at industries and branches which are somewhat related to NGOs the biggest source of inspiration is the healthcare industry. Similarities lie in being an industry not fully focused on making profits, not under as much pressure to optimize and caring more about the "customers" than most commercial industries. An example of a medical research being relevant to our thesis is "Smooth Bayesian network model for the prediction of future high-cost patients with COPD" written by Lin et. al. (2019). They predict chronic obstructive pulmonary disease or COPD spending,

and while the framework they use is not structured in a way that we can use it in our case, it can still serve as inspiration. A problem they have with their data is the same as ours, and this makes it an interesting research to consider their methods. One of the models they use is a type of K-Nearest Neighbors model, a model we also test, and which is explained in one of this chapter's later sections. The benchmark models used in this study are also compared to other benchmark models, and considered as candidates for this thesis.

Looking further than healthcare we looked at predicting project-based spending. This led us to studies looking at large IT and building projects, but these are not comparable to the type, duration and frequency of the projects the NGO does. These predictions were one-off predictions requiring a large number of usually very specific inputs. We focus more on looking at the problems of our data and a second study we found when looking on how to deal with sparse data is "Improved Web Service Recommendation via Exploiting Location and QoS Information" by Gonsalves and Patil (2016). This paper looks at connecting internet users with the optimal provider in the optimal time. This means that it is important for the model to be quick to make a recommendation but also be accurate. Although there are few parallels between recommending a web service and predicting project spending, some of the main challenges are similar, and as such we will investigate whether their models can also deal with our problems. For their research they used a K-Nearest Neighbors algorithm with a Support Vector Machine. The fact that they also used a K-Nearest Neighbors component gives use more reason to believe this model may work well for our case. We will investigate whether a Support Vector Machine variant suitable for our problem is able to predict our data well.

## 3.2 General Machine Learning introduction

This section focusses on the what Machine Learning is, and will explain some of the core principles. This will be used to give an understanding of ML, and these concepts will also be used to argue why certain models and algorithms are or are not suited for the prediction of Field spending. First an overview of some of the different types of Machine Learning will be given, followed by an explanation why data splitting is used. The final three sections explain the concepts of features, over- and underfitting and heteroskedasticity respectively.

### 3.2.1 Types of Machine Learning

There are four types of Machine Learning which will be discussed here to create an understanding of what ML can do, these are Supervised Learning, Unsupervised Learning, Semi-Supervised Learning and Reinforcement Learning. There are more types of Machine Learning, the ones explained here are used more commonly and more relevant to the research goal. The first type is the kind used in this thesis which is Supervised Learning (SL). Here the ML model is given data with specific targets, which are called labels, and the ML model will try to find a model configuration where the prediction of the model is as close to the label as possible. An example of this is the MNIST dataset, one of the most iconic ML datasets which use this type of learning, which contains 70,000 images of handwritten numbers together with the number the image is supposed to represent. The goal is to correctly predict the number written on the image. As of the writing of this chapter the best result reached on the 10,000 test images of the MNIST dataset is an error rate of 0.21%. (MNIST database, n.d.)

A different type of learning is Unsupervised Learning. While Supervised Learning had a label for each datapoint, Unsupervised Learning does not try to reach specific values, instead it is aimed at organizing data. Applications of Unsupervised Learning include recommendations of video services, where the ML model looks at the similarity between videos, and anomaly detection, where fraud prevention looks at patterns which do not fit normal behavior.

A mix between these two is Semi-Supervised Learning. This is often used when there are not enough datapoints with labels to allow for Supervised Learning, but when there are a large number of datapoints which do not have labels. The similarity between the labeled and unlabeled datapoints used to assign the most likely label to the unlabeled datapoint, and after all unlabeled datapoints have been labeled the entire set is used to train the predictive model. A possible application for this is with medical images such as MRI's or CAT scans, where an image may or may not contain a tumor or other medical condition. Since analyzing these pictures is expensive due to the amount of knowledge needed creating large databases with pictures and labels is too expensive. By using Semi-Supervised Learning a model can be made which, while not as good as a Supervised model, can give predictions on the healthiness of the patient. If the model is not perfect but still has enough good results it could be used to flag pictures and areas which are likely to contain tumors, with an expert checking if this is actually the case. (Salian, 2019)

The final type of Machine Learning is considerably different from the previous ones. This type is called Reinforcement Learning and is about which action is optimal at what time. Here the model is referred to as an agent, which is active in an environment. The agent observes the environment and decides what action to take. DeepMind's AlphaGo Zero and AlphaZero were trained using Reinforcement learning to beat humans and other model in the games of Go, Chess and Shogi, and is one of the famous examples of this kind of Machine Learning. After taking an action the agent may gets a reward or penalty based on what happened in the environment. When the environment is updated the agent observes the new situation and decides on another action. A common application of Reinforcement Learning is in (video) games. The chess board or game is the environment, and the valid moves for each piece, or each button which may be pressed are the actions the agent can choose from. Rewards may be gained or lost based on the point value of a chess piece or on the score of a game. The agent learns what the expected results will be if a certain action is taken in a certain environment state. An important part of this expected result is that it also includes future rewards. This way the agent is able to think about the long-term consequences of the action, and is not just choosing the action with the most immediate gain. While this type of ML is not limited to games, the models do require some freedom in experimenting how certain actions will affect future returns. While a self-driving car could be trained with just a Reinforcement Learning where rewards are based on fuel usage and time taken and penalties for damages to the car, having a model learn by crashing cars in real traffic is not ethical or financially sound. If the model can train in a realistically simulated environment this method of ML can be used to teach a model to drive a car. Just like the other types of ML Reinforcement Learning has certain areas where it fits naturally, while with some extra steps it can be used in some other areas. (Silver, 2015)

As already mentioned in the part discussing Supervised Learning, the problem this thesis is trying to

solve is one of Supervised Learning, as all used datapoints have labels, and it fits the optimization goal of SL. It does not fit either the Unsupervised or Semi-Supervised Learning variants due to data structure and the problem does not have an environment or action space as seen in Reinforcement Learning so this cannot be used either.

### 3.2.2 Input data

Machine learning uses some more uncommon terminology, and features is one of them. In machine learning a feature is an input variable which has some relation to the desired output. These features are intended to allow a model to find the best or most suitable outcome. In a model where the likelihood of side-effects for a patient is estimated features such as age, sex, occupation, other medication and dosage are likely to be included. When working with text as features, such as an occupation feature, most models cannot use this as an input. Instead numbers are used to represent the occupation. There are different ways to do this, a common one will be discussed in chapter 3.3.1. Choosing good features is an important part of building a model, as having features which do not relate to the output or are very loosely related will slow down training. It may even worsen the performance of a model, and can increase overfitting, which is the topic of the next section.

### 3.2.3 Correctly finding patterns in data

A major issue with models is over- and sometimes underfitting. Overfitting occurs when a model finds patterns in the examples it was trained on, which do not generalize to reality. If this happens it will be very reliable on the training data but will perform poorly on new data. Figure 4 and Figure 3 are used to illustrate these two concepts. The two figures attempt to find the function which best approximates a set of points.

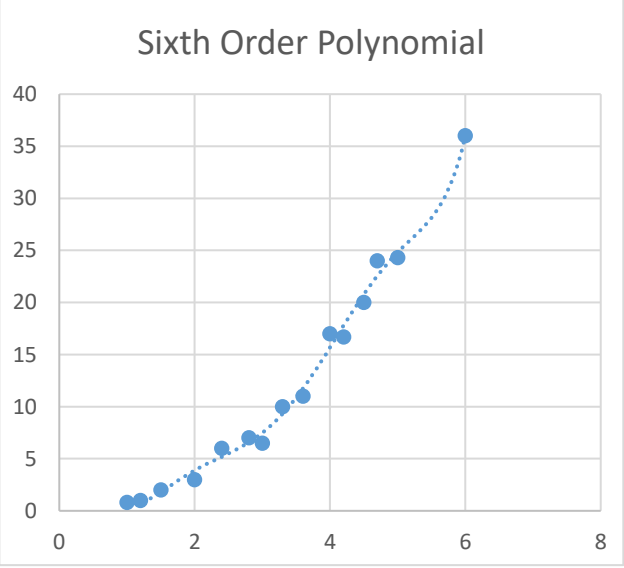


*Figure 4 - Sixth order polynomial example, which follows the datapoints more closely, but may be overfit*

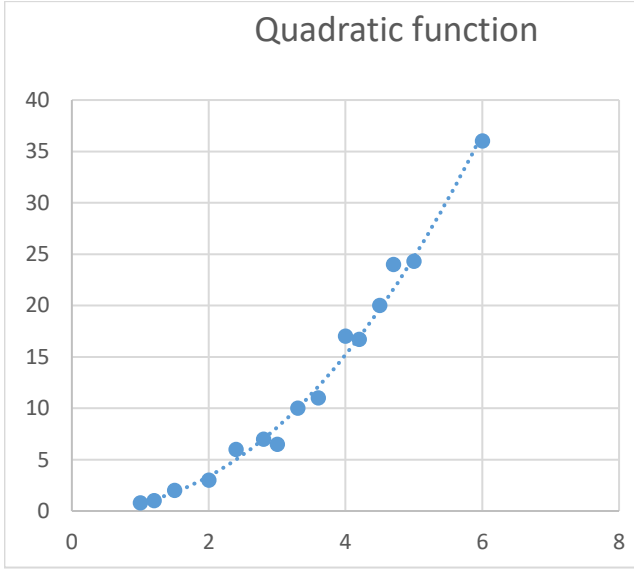


*Figure 3 - Quadratic function example, which follows the general trend through the datapoints, but might be underfit*

While in this case Figure 4 is not necessarily overfitted, the shape of the trend line between 5 and 6 does suggest that this might be the case and extra steps to ensure this is not the case should be

taken. Because the goal of Machine Learning models is to be able to make accurate predictions or classifications about data it has not seen before it is important to prevent overfitting on the training data. The opposite of this is underfitting, where the model is kept too general when more information could have been gained from the available training data. In the figures shown this would imply the opposite from overfitting, with the sixth order polynomial being closer to the actual function and the simpler quadratic function missing out on using some of the available information.

The Bias-Variance trade-off is another name for the same problem, and looks the problem from a more technical point of view. The error of a model is built up in different parts. Part of the error is from the inaccuracies from model, and part is independent of the model and is the result of noise in data. This noise is a combination of small influences which cannot be reasonably accounted for. The error from the model can be split in two parts, one for bias from the model and one for variance from the noise. When variance is minimized bias increases, as the model becomes rigid and is underfit while a model with low bias has a higher variance as it becomes too flexible and overfits (Bishop, 2006).

When overfitting is suspected this is commonly reduced by introducing weights penalties. As small weights tend to cause graphs similar to Figure 3 and larger weights more often lead to graphs like Figure 4, and sometimes more extreme, a solution to preventing more erratic predictions is to limit the (negative) growth of weights during training. Two methods of limiting the growth of factors are Lasso regression and Ridge regression, also known as L1 regularization and L2 regularization. Here L1 reduces the weights by setting a limit to the sum of absolute weights, while L2 limits the sum of squared weights. The parameter associated with these regularization terms balances how constricting the regularization is versus how much the model is allowed to freely learn from the training data. Since we do not end up using this method for any of our final models we will not go into further detail on the working of these methods of reducing overfitting (Tibshirani, 1996).

### 3.2.4 Ensuring validity of model performance

In order to make sure the estimated performance of a new model is good, data is often split in different parts, the most basic split is to split into training data and validation data. The training data is used to find the best values for the parameters of the model, while the validation data is used after these parameters have been found to see how well the model performs. This is done to prevent overfitting. Sometimes the training data is reduced further so there are three data sets: the validation set, the testing set and the training set. This can be used if the hyper-parameters of the model need to be determined. Here the validation and training set serve the same purpose as before. After the model is trained on the training data the performance is measured on the testing data. This is done for each hyper-parameter configuration which is tested. After the best settings have been found the model is trained using those settings with both the training and testing data. Finally, the performance of this final model is estimated by looking at the performance on the validation set (Bishop, 2006).

### 3.2.5 Dealing with unequal variance of the error

For most Machine Learning models, as well as for analysis of models and their output, a number of assumptions are made with regards to the training data and output. These can be simplifications or restrictions necessary for the statistical and mathematical theories and techniques used (by the

model). When making a prediction model the datapoints, $Y_i$ $(i = 1, .., n)$, are expected to come from a combination of two parts $Y_i = X_i\beta_0 + \varepsilon_i$. In this equation $X_i$ represents features of the model, which are transformed by operations referred to as $\beta_0$, which is what the model is trying to learn. $\varepsilon_i$ is the unobservable part often referred to as noise, which is not predictable for a given datapoint. This unobservable part is assumed to be independent not (necessarily) identically distributed (i.n.i.d.) from the predictable part. Heteroskedasticity is concerned with the unobservable part of the equation. While the unobservable part cannot be predicted for a single point, there are properties that can be found about the distribution(s) the unobservable part comes from. Heteroskedasticity means that within a population there are several sub-populations which have different variances. As a number of statistical test and operations assume equal variance it is important to keep in mind the possibility of heteroscedasticity when analyzing the performance of a model. One way to test for heteroskedasticity is to use a White-test, which will be used later for the final model, and which has been used as a main source for this model. Combatting heteroskedasticity can be done by transforming what the model predicts to make sure that the predictions are well scaled in relation to each other. This will hopefully get unify the population and get rid of all sub-populations (White, 1980).

## 3.3 Machine Learning Techniques

In this section various techniques and algorithms will be discussed to deal with some of the problems and concepts discussed in chapter 3.2, and deal with imperfect data.

### 3.3.1 Using categorical features in Machine Learning

One-hot encoding is an important tool in processing data so it can be used in Machine Learning. Models in ML are limited to working with numbers, and requires all input to be in numbers. If the input is the budgeted amount this is fine, and for black and white pictures the gray-scale value of each pixel can be used as input. When working with a feature which is a category, such as the country of a project, this does not work. Even if they are represented as country 1, country 2 ... country 10, taking the country number will still not work, as there is no scale or set relation between the numbers. Instead of the single country feature, or another categorical feature, one feature is added to the input for each unique value of the category. In the country example 10 features, e.g. country1, country2, etc., will be added, where each feature represents 1 country. The value of the feature is 1 if the project is in that specific country and 0 otherwise. The advantage of One-hot encoding is that by giving each unique value a feature, it allows the ML model to learn the interactions of each feature with the other features. The downside is the additional computing strain, as more features result in more weights which need to be updated, and more memory which is needed to train the model (Hale, 2018).

### 3.3.2 Feature selection and Principal Component Analysis

As mentioned in chapter 3.2.2 it is important to select the right features for a model, as it increases performance, robustness and reduces training time. Perhaps the most obvious way is to check each possible combination of features to include in the model, and save the combination with the most predictive ability. However, this is not always feasible as the number of combinations of features increases exponentially, and as such the time needed to test all these combinations increases exponentially as well. A manual check of features is always a good idea, to check if certain features should not have any predictive power, such an ID, and if there are features with high correlation. Once the unnecessary features are removed there are algorithms to find the best set of features.

Examples are Forward- and Backward Search, with Forward Search the model starts with no features and adds one during each iteration until the performance no longer improves or a preset number of features have been selected. Backward Search is the opposite, where the algorithm starts with all features and removes one at a time. A more complex way of feature selection is the Floating Search methods. This contains two methods, one for a Forward Search and one for a Backwards Search. This method will also remove previously added features on the Forward Search if it finds a feature to have too little predictive power after adding a new feature, and will add back features in the Backward Search if after removing a feature one of the removed features out be useful enough in the new configuration (Pupil, Novovičová, & Kittler, 1994). Different adaptations to this algorithm have been developed on this basis.

However if there are too many features a different approach can be taken as well. One of these is the Principal Component Analysis. This method projects data points on a lower dimension while maintaining as much variance as possible. A simple example of a dimensionality reduction from two to one can be found in Figure 5.

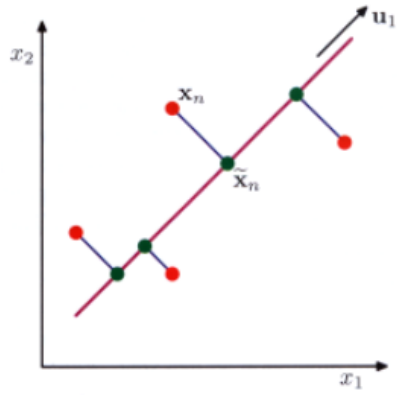


*Figure 5 - Principal Component Analysis (from Bishop, 2006, p. 561)*

PCA supports any number of starting dimensions and final dimensions as long is these are fewer than the starting amount. The first component tries to capture as much variance in the data as possible. Since ML models use the variance between datapoints to learn, the goal is to retain as much variance as possible. Every component after tries to capture as much variance as possible under the restriction that the direction of the component is orthogonal to the previous components. Because they are orthogonal to each other the different components are uncorrelated, so each feature adds a completely new piece of information (Bishop, 2006).

When considering predictions in a series predicting one step back is the conjugate reverse of a one step forward prediction. A model will learn the difference between two steps, therefore predicting one step forward is almost the same as predicting one step backwards. This relationship has been used to build an algorithm which can efficiently calculate certain equations involving a particular type of matrix. The relationship will be used later in the thesis as a basis for an analysis (Theodoridis, 2015).

### 3.3.3 Hold-Out

If there are enough datapoints it is common practice in Supervised ML to use Hold-Out where the available data is split into two parts, part A which is the training data, and part B which is validation data. First good settings for the model are found using the training data, after which the model is trained using the same training data. The ability of the model trained on the training data to correctly predict the validation data is used to estimate the performance of the model. When a final model has been selected by looking at the estimated performance, the final model is then trained on the complete data set containing both part A and B. It is important that the training data set contains enough datapoints to cover all likely possibilities, and the testing set should be representative of reality, which comes down to needing to be large enough so outliers do not significantly influence the estimated performance. Since this is often not very feasible the next section discusses a different approach (Bishop, 2006).

### 3.3.4 K-fold cross-validation

A common way to test for overfitting and to get a prediction as to the performance is by using K-fold cross validation, where K is an integer chosen by the programmer(s). This technique is used when a large number of model configurations are compared or if there are not enough datapoints to use Hold-Out. If one validation set is used to compare these different models' configurations (also known as hyper-parameters) there is a larger risk of overfitting on the validation set, especially in the cases of limited datapoints or many models which are tested. With K-fold cross-validation the rest of the data is split into K different parts of equal size. This means 5-fold cross-validation has 1 validation set, and 5 data sets of equal size, and every datapoint is assigned to a single data set. When testing different model configurations with the 5-fold cross-validation the model is trained five times. During each training run one of the data parts is used as test set and the 4 other data sets are used to train the model. The average performance of the model during the five runs is used to estimate the performance of the particular model configuration. By using the whole training set in different runs prevents the model from overfitting on a single test set. Furthermore, if the test set is small the performance estimation will be noisy, where things such as outliers in the test set or an unusual large amount of similar datapoints give a wrong performance estimation. Some rules of thumb exist to find a suitable number for K, but there are no set rules. Increasing the number of data set parts should decrease the noise, but it also adds increases computation time. If there are a large number of model configurations which need to be tested computational time become a problem (Bishop, 2006). When comparing results or reporting findings K-fold cross-validation is preferred over Hold-Out, as reliability of the results improves, and more statistical insights can be found. A certain type of K-fold is the stratified K-fold whereby the goal is to spread the prediction goals over the dataset as evenly as possible.

### 3.3.5 Genetic Algorithm used to tune model parameters

Models and in particular Neural Networks (explained in chapter 3.4.2) have a considerable number of parameters which need to be configured. In this thesis a Genetic Algorithm (GA) is used to find good and appropriate values for these parameters. A GA starts with a population of different chromosomes, in the case of this thesis each chromosome is one model configuration. The algorithm has two basic steps: Select the parents from the chromosomes, where the probability of selecting a chromosome increases based on the fitness of the chromosome. Fitness represents the

quality of the chromosome; in this thesis' case a lower prediction error increases the fitness of a chromosome. After selecting two parents some form of crossover and mutation happens where the offspring is generated based on parents. Crossover combines the genes from the parents, while mutation changes (part) of the offspring further.

This thesis uses "Tuning of the Structure and Parameters of Neural Networks using an Improved Genetic Algorithm" from Leung et. al. (2003) for the selection of the parameters of the NN. The rest of this chapter will describe their implementation of GA to find the right parameters for an NN. The first step of choosing the parents is simple process where each chromosome $p_i$ is assigned a probability $q_i$ of being selected which is done by dividing the fitness of a chromosome $f(p_i)$ by the fitness of all chromosome:

$$q_i = \frac{f(p_i)}{\sum_{j=1}^{pop\ size} f(p_j)}$$

This means that the sum of all probabilities is equal to 1, and that a chromosome which has a fitness which is twice as good as second chromosome is twice as likely to be selected in comparison to the second chromosome.

After two chromosomes have been selected this way the crossover operation starts by generating four different offspring according to the operations found in Equation 1.

$$\begin{aligned} os^1 &= \frac{p_1 + p_2}{2} \\ os^2 &= \frac{(p_{max} + p_{min})(1 - w) + (p_1 + p_2)w}{2} \\ os^3 &= p_{max}(1 - w) + \max(p_1, p_2) * w \\ os^4 &= p_{min}(1 - w) + \min(p_1, p_2) * w \end{aligned}$$

*Equation 1 – Leung et. al. Genetic Algorithm crossover operations*

Where $w$ is a number between 0 and 1 which is to be determined by the user, $p_{max}/p_{min}$ are vectors containing the maximum and minimum allowed values for each parameter which is being optimized. $\mathrm{Max}(p_1, p_2)$ is the vector which contains the maximum value for each parameter found in the parent chromosomes, $\min(p_1, p_2)$ is similar but contains the minimum value per parameter instead. The first two of these offspring move towards the average values of the parameters, while the third and fourth offspring move towards the extreme values.

After generating these four offspring they are evaluated for their fitness. The offspring with the best fitness is selected to continue to the mutation operation. Leung et. al. generate new crossover offspring to form three different final offspring. The first (crossover) offspring has just one of its parameters changed by a random amount, such that the new value is within the allowed range. The second offspring has a random amount of chromosomes changed in the same way as with the first, while the third offspring has all of its chromosomes changed by a random amount within the allowed range. These three final offspring are evaluated on their fitness, and one of them is chosen to replace the chromosome with the lowest fitness in the population. The chromosome is chosen by

randomly generation a value between 0 and 1 and comparing the value to a user defined $p_a$ which is also between 0 and 1. If the randomly generated number is less than $p_a$ the offspring with the best fitness value is selected, while the offspring with the worst fitness is selected if the generated number is more than $p_a$. By accepting the worst of the three offspring the chance of converging on a local optimum is reduced, consequently improving the chances to find or approach the global optimum (Leung, Lam, Ling, & Tam, 2003).

## 3.4 Models

This section discusses some of the model families and some particular we will test to use to predict the spending of field projects. For different reasons each model is interesting for our thesis. The models build on the concepts described in previous sections of this chapter while introducing a few new concepts which are typical or unique to the model.

### 3.4.1 K-Nearest Neighbors

The concept of K-Nearest Neighbors (K-nn) is: can we find the K most similar datapoints to a new datapoint and use them to make predictions about the new datapoint, where K is an integer denoting how many similar datapoints are considered? K-nn can be used both when trying to predict a class or when predicting a continuous value. When trying to identify a plant species the most common type among the K nearest plants will be the predicted species, and for car price predictions an average of the closest other cars is taken. The choice of parameter K is free, with k=1 meaning that just the closest observation is considered, and k=5 the five closest observations (Altman, 1992).

There are no set ways to calculate the distance when predicting continuous values, Euclidian distance is an option for continuous variables and Hamming distance on for discrete variables. This way the datapoints which are closer to the unknown datapoints than others have weights assigned so that datapoints closer to the new datapoint are more important. The choice of how to calculate distance is up to the programmer, and this choice is very important as it can heavily influence which datapoint is considered similar. One of the weaknesses of K-nn is that it is sensitive to the structure of the data. When the sale of ice cream is predicted and temperature is one of the features there might be different nearest neighbors depending on if Celsius or Fahrenheit is used, as the difference of 5 degrees between 30 and 35 degrees Celsius is one a of 9 degrees with the equivalent 86 and 95 degrees Fahrenheit. K-nn will be used as a benchmarking model as it is able to find less complex patterns but struggles to find very deep patterns, which can give nice insights if there is a need for more intricate models or if simpler model suffices. It is also included as it has worked well as part of a larger model for Lin et. al. (2019) with their model which has some similar data structure problems. Furthermore it was used by Gonsalves and Patil (2016) who experienced similar data problems too.

### 3.4.2 Standard Neural Networks

Neural Networks (NN) are a family of models which are able to find nonlinear relations between the input and output. As it is able to capture nonlinear relations it is able to find more complex patterns than model such as K-nn and linear regression. While NNs are inspired by nature, the ones discussed here are not representations of the brain. Bishop (2006) notes the following: “From the perspective of practical applications of pattern recognition, however, biological realism would impose entirely unnecessary constraints.” A basic NN diagram from Bishop can be found in Figure 6.

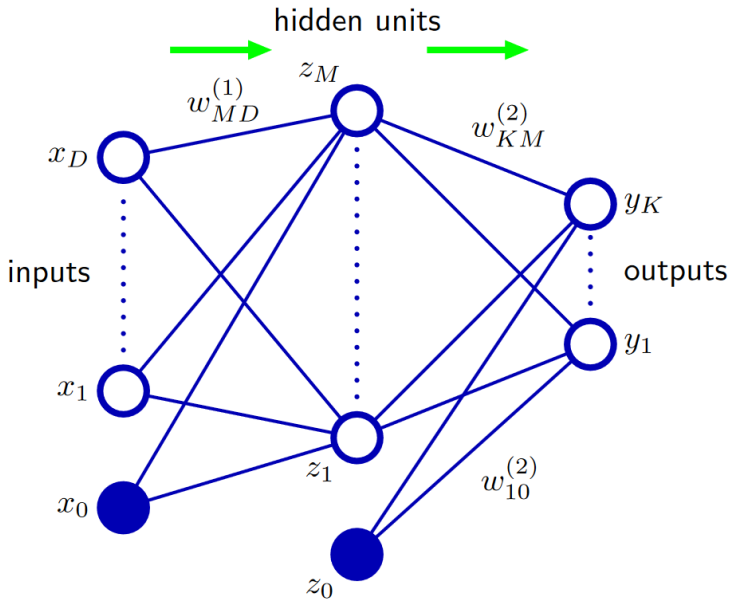


*Figure 6 - Neural Network diagram (from Bishop, 2006, p.228)*

The nodes represent the input, hidden units and outputs. If there are ten features, the model would have eleven nodes, $x_1, x_2, x_3 \ldots x_{10}$, one for each feature, and one node $x_0$ which is called the bias (which will be explained later). The next nodes in the model are the hidden nodes. These are activated by one of a variety of activation functions, although they share that each one is non-linear and is differentiable. This last property is crucial for the learning process of the NN. The output layer transforms the data further through another activation function which may be different from the hidden layer. The activation function of the output layer is determined by the data and the structure of the target values. If the NN needs to differentiate between two different classes, e.g. healthy or infected, the output function is transformed using a logistic sigmoid function, which can be found in Figure 7. This function outputs values between 0 and 1 with more extreme values for $z$ (the input) getting very close to the 0 or 1. During the prediction, the value of every node after the input layer is determined by summing up the previous nodes' values multiplied with the weights connecting the two nodes. After summing these values the summed value is put in the activation function to produce the final value for a node.

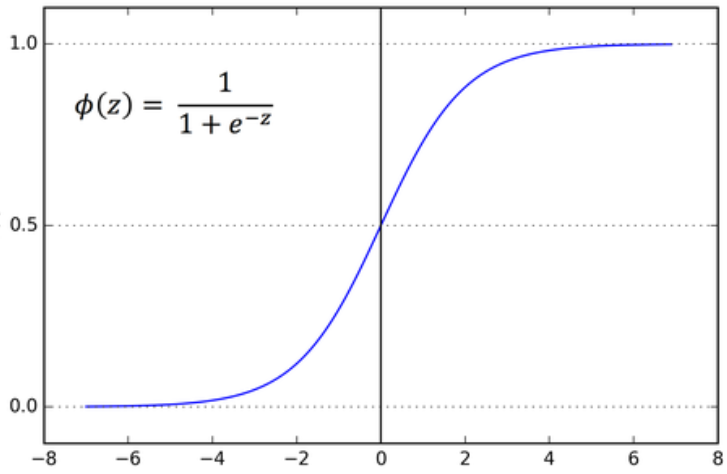


*Figure 7 - Sigmoid Function (Sharma, 2017)*

By introducing a bias term the node output can be shifted to the left or right, and this is learned by the model as it is trained. Which activation function to use for hidden layers is partially dependent on the data structure and desired output, and partially a trade-off between factors such as computing speed, normalization and convergence. Choosing the most suitable activation function for the hidden layer remains mostly a process of trial and error where selecting functions which may work are combined with the outcomes of experiments.

Connecting the different nodes are weights. These weights and the biases for each layer are what are changed when training the model. In the notation $w_{M\,2}^{(1)}$ the w is for weight, the superscript indicates that it connects from the first layer, the right subscript shows it connects from node 2, and the subscript to the left is shows that it is connected to node M on the next level. Typically each node on one layer is connected to each node on the next layer with unique weights for each connection. This is not always the case, which we will explain for these models. The weight of the bias to nodes is unique in that it is always 1, and instead the bias value is changed (Bishop, 2006).

There is no set way of determining the number of nodes in the hidden layer, there are some rules of thumb but no agreed upon algorithm or heuristic has been developed. This is another setting which is also part of the trial-and-error process of finding good parameters for a Neural Network. There are some heuristics which are used commonly, such as Genetic Algorithms, which we will use and have discussed in chapter 3.3.5. Although in theory an NN with one hidden layer with enough neurons and enough training data can approximate any function to any desired accuracy, this is becomes computationally expensive and relies on enough training data. Therefore having a model of one layer becomes practically impossible if the complexity of the prediction target increases. Due to the large amount of possible model configurations, even more due to the multiple layers, it is common to use a heuristic instead of a manual search.

Deep Neural Networks are different from NN by having more than one hidden layer. This can allow the Deep NN to find complex patterns more easily than a single layer NN, although more layers will not necessarily improve performance. As mentioned, this is another parameter of the NN which is

determined by rules-of-thumb, trial-and-error and heuristics although a popular sentiment is that most current problems can be solved with a single hidden layer, so without a Deep NN. When considering the number of hidden nodes and number of layers having too many will increase the likelihood of overfitting as the model has overcapacity to train on the training data. More nodes also increases the number of weights to update, and between each layer gradient calculations need to be made which means that training the model takes longer as well. What the gradient is will be explained in the next paragraph.

As mentioned before, the property of the activation function of being differentiable is very important for training. In a process called backpropagation the gradient of the error is calculated and propagated throughout the NN to update the weights. The gradient consists of vectors for each point of a function, with one vector consisting of the partial derivatives of each variable. The gradient of a graph is the direction of the downwards slope of a graph, and when minimizing the error some kind of gradient descent is used, as the goal is to descent to a minimal error value. Calculating the gradient uses differentiation which is combined with the mathematical chain rule to find how much and which way the weights need to be updated when training an NN. The true gradient of the error value cannot be known in practice as it requires a perfectly representative dataset. Therefore an estimation needs to be made. When updating a learning rate is used to change how quickly the model adjusts to the estimated gradient, which is denoted by α which ranges $0 \leq \alpha \leq 1$. A small value for the learning rate means that the model learns very slowly, while for a value which is too large the adjustment overshoots (Bishop, 2016).

There are several ways to estimate the error gradient and these algorithms are referred to as optimizers. Some of these automatically adjust the learning rate, while others have one learning rate which is given by the programmer. One method is known as by several names: on-line gradient descent, sequential gradient descent or stochastic gradient descent. This method randomly takes one datapoint, and updates the weights based on the error of this one datapoint. Each datapoint is selected once until all datapoints have been selected. One iteration over all datapoints is called an epoch. The training is continued either until a maximum number of epochs has been reached, performance is good enough, or when no improvement in accuracy has occurred over one or several epochs. Shuffling the datapoints after each epoch can prevent cyclic adjustments which reduce the learning ability if left unattended. Batch Stochastic Gradient Descent uses batches instead of single datapoints to find the error, while Gradient Descent uses the error of the entire dataset. Gradient Descent requires a lot of computations as it considers the entire set each update and gets stuck in local optima, so the other forms are used more often. Root Mean Square Propagation (RMSprop) is an unpublished gradient descent optimization algorithm, but is well known and integrated in ML libraries such as Keras. RMSprop keeps track of a moving average of the previous gradients, and uses this to change to learning rate (Frumkin, 2019).

There is one final part which is needed to train an NN which is an error function. The error function is used during training to shows how good or bad the prediction is compared to the label. Since only Supervised Learning has labels for all the datapoints, this is a unique feature for SL. This value is used when updating weights to see how much they need to be changed. For NNs there are a variety

of error functions, however where many of the previously explained parameters of NNs follow rules-of-thumb and trial-and-error, this is less so the case for error functions. There are clear reasons when certain error functions are appropriate, which should become somewhat apparent after explaining some of the error functions. When looking at the difference between the prediction and actual value the difference between the two values can be taken as the error. This is called the Mean Absolute Error and would imply that for example being off by 20.000 dollars on a 20.000 dollars actual value is just as bad as being of 20.000 on an actual of 400.000. By taking the percentage difference between the prediction and label values the first case is considered worse than the second case, and will result in a larger change in weights. This second way of measuring errors is called the Mean Absolute Percentage Error. The loss function which is most appropriate for a model is the result of the type of the problem and the priorities set for accuracy. A third error function is the Mean Squared Error, which squares the error and as such penalizes being far off target. It still considers being wrong by 20.000 dollars on a 20.000 dollars budget is just as bad as being of 20.000 on budget of 400.000, however being off by 20.000 compared to 30.000 is much worse: $\frac{30000^2}{20000^2} = 2.25$, so more than twice as bad. This error function therefore gives priority to being wrong roughly the same amount for each prediction, as larger errors bring larger weight updates.

As has been discussed in this section on Neural Networks there is a considerable number of parameters which have to be found and tweaked for a good configuration. This is why the previously discussed Genetic Algorithms (chapter 3.3.5) have been used to find a good configuration. Finding them is one of the largest downsides of NN as it can take a lot of time to find a good model. Because of the power and versatility NNs can be worth the effort but if time is more constraint or if the problem is relatively simple other models are a better choice.

### 3.4.3 Neural Network variants

This section will discuss two Neural Network variants, which are interesting for different reasons. The first variant is the Dropout Neural Network, which is one of the techniques used to combat overfitting in Neural Networks. This has been patented by Google (United States Patent No. US9406017B2, 2016), although it is publicly available under the Apache License version 2.0, which means that is can be used without any copyright.

The method works by randomly dropping nodes and their connections during training, the final result of which can be seen in Figure 8. This means that during each training update only a part of the nodes is active and updated. After training the average of all possible permutations of activation or dropping nodes in the network would be the best prediction for the true value. However for most networks this would take far too long to calculate. This sum can be approximated by taking the full network with scaled down weights, which is directly proportional to the percentage of nodes which are trained during each training case.

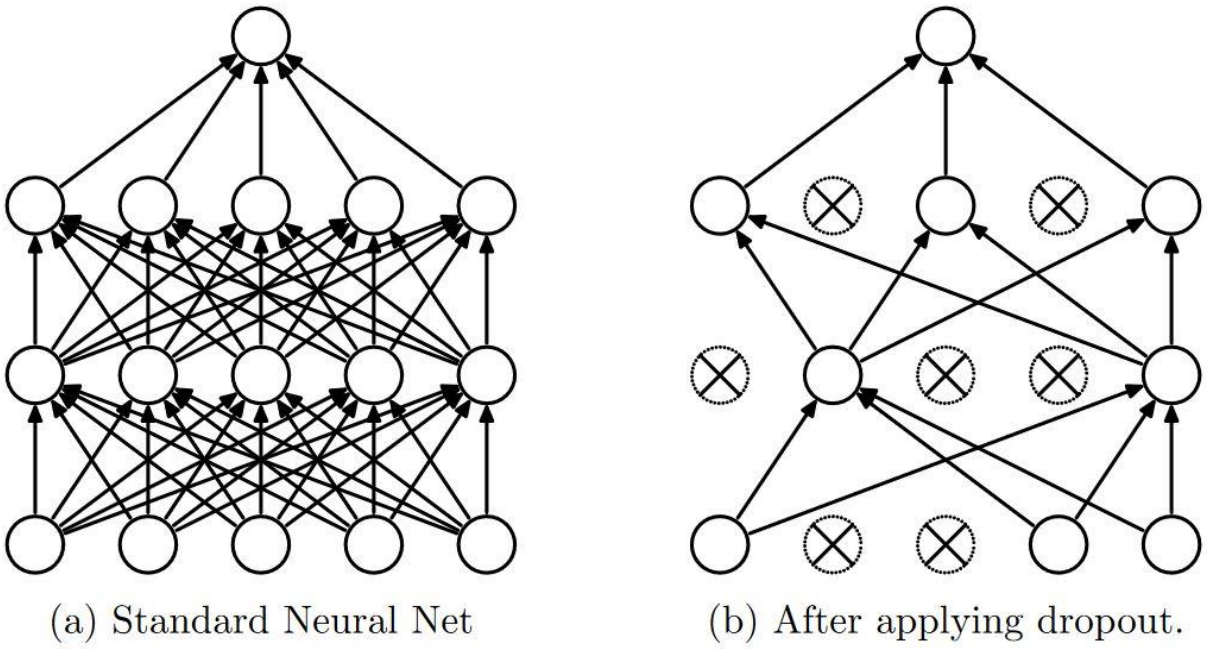


*Figure 8 - Standard Neural Network compared to a Dropout Neural Network (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014, p. 1930)*

There are two main advantages to dropout: The first is that by making the presence of any hidden node unreliable the co-adaptation is broken up, which means that where previously with standard backpropagation some hidden units would develop highly correlated behavior, this is broken up with dropout. Secondly, performance is increased due to the trained NN being a combination of many different models, without having to fully train each model. If unlimited computation resources ware available the best way to prevent overfitting would be to average the predictions of all possible settings of the parameters, weighting each setting by the probability of the datapoint being of each model's respective training data. This is not (yet) feasible but a dropout NN is a way to approach this while still being computationally feasible. The biggest downside is that training takes longer, typically 2-3 times longer according to Srivastava, Hinton, Krizhevsky, Sutskever and Salakhutdinov, a major cause of this is believed to be the noisy parameter updates (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

Normal Neural Networks are considered dense, which means that each layer has a set number of nodes which are each connected to all the nodes in the previous and next layers. To reduce the strain on memory and processing power Sparse Neural Networks (Sparse NNs) are being developed. The goal when developing new Sparse NN algorithms is to reduce the number of parameters as much as possible while performance remains (nearly) the same. Zhu and Gupta have found in their experiments when comparing smaller dense NN with larger sparse NN with identical memory footprint that larger Sparse NN had better performances.

The method they used gradually increases the number of weights, which are connecting nodes in subsequent layers, set to zero, meaning the two nodes do not interact with each other anymore.

This is done by sorting all the absolute weights connecting two layers from small to large and changing a portion of the smallest nonzero weights to zero. These weights are also exempted from updates reducing calculations needed during updates, thus causing the weights to remain zero.

The experiments by Zhu and Gupta suggest there is an optimal range to which dense models can be compressed into sparse models. Their results suggest starting at a model between 5 and 10 times the desired size and pruning it until it reached the desired number of parameters. One peculiar phenomenon Zhu and Gupta came across during training is that at times the model may have a near-catastrophic degradation of performance, which recovers almost as quickly with continued training. This effect was more pronounced in the models trained to trained to have higher sparsity.

While the model may have only one tenth of the parameters this unfortunately does not translate in a 90% reduction in necessary memory, as the sparse matrix storage and computations are not as efficient. Nevertheless, in the experiments the performance of sparse networks was better than the dense networks with equivalent memory usages. The performance gap can further increase when deep learning architectures which are better at handling sparse storage and computations, and Zhu and Gupta think their results give a further impulse to develop this (Zhu & Gupta, 2017).

For our thesis we will use three of the Neural Network variants:

1. The Standard Neural Network, which may be a Deep Neural Network, or a one-layer NN, which has the standard architecture.
2. The Dropout Neural Network, which may be a Deep Neural Network, or a one-layer NN, which has a standard architecture, but during each training step it will randomly drop some nodes and their connections and update without taking them into consideration.
3. The Sparse Neural Network, which may be a Deep Neural Network, or a one-layer NN, which has the standard architecture before training, but during training it will permanently stop some nodes from working by setting their weights to 0

### 3.4.4 (Gradient Boosted) Decision Trees

In this section we will discuss a specific type of Decision Tree. A Decision Tree splits up predictions based on certain conditions, such as: is the budget < 5000, is it in the interval [5000,15000] or is it more than 15000. The prediction as to the final spending prediction could be 3000, 8650 and 14000, this example is demonstrated at Figure 9. While the example decision tree has only 1 step a decision tree may have many of steps where each time the tree splits on different criteria. The type of decision tree we will be using is Gradient Tree Boosting, which consists of two parts, the Decision Tree, and the gradient boosting algorithm to design this regression tree. The Decision Tree is the model family, while gradient boosting is a general technique, although for convenience we explain them together, rather than split up over two sections. Boosting is explained by Friedman as having an initial guess $\boldsymbol{p}_0$, where successive increments $\{\boldsymbol{p}_m\}_1^m$ ("steps" or "boosts) are combined to form the final predictor of $\boldsymbol{P}^* = \sum_{m=0}^{M} p_m$. Each new step is based on the sequence of the preceding steps, while the computation of each new steps is defined by the optimization method, which is gradient descent in this case. As mentioned in chapter 3.4.2 on neural network the gradient descent means that the algorithm tries to find the optimum by finding where the gradient of the error is to find how the error can be decreased the quickest at that particular place. As with Neural Networks

there is not a set error function, and the same range that can be also be used for Gradient Tree Boosting, such as mean square error or mean absolute error.

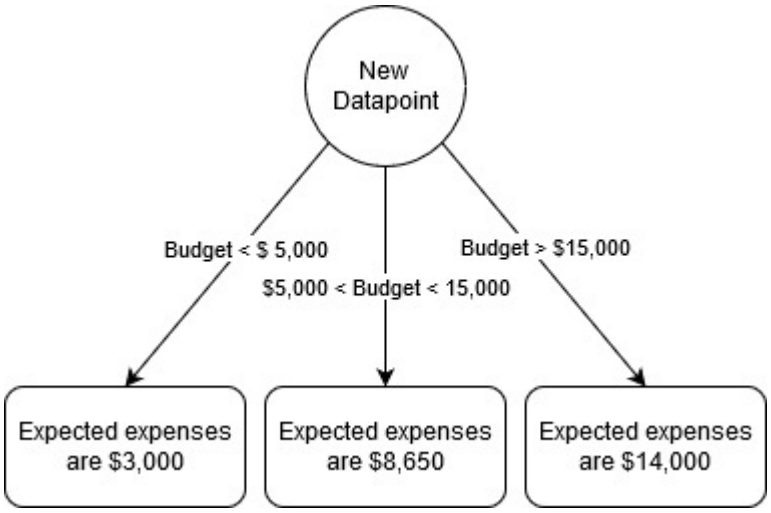


*Figure 9 - Basic Decision Tree example*

The intuition to the combined method of Gradient Tree Boosting is to create a model which takes weak learners (predictions only slightly better than random), in the form of decision trees, and combines them to make a good prediction. The new decision tree which is added at each step tries to make up for the shortcomings of the previous combined model.
A downside of using Gradient Tree Boosting for regression problems is that it becomes a step function as the decision trees use single values, which do not scale with input. An example of this would be two projects with the same data except one having a budget of 15.000 USD and the other 14.000 USD. It can happen that the difference between these values results in the same final expected spending of 13.800 USD, as the path could be the same for both projects. While this could be addressed by adding a new split or new tree, this does not work for unseen data and results in a loss of generalization (Friedman, 2001). Gradient Tree Boosting will be used as one of the benchmarking algorithms, although it may not be very suited to this problem, it is a commonly used algorithm and can still serve to give insights into the data.

### 3.4.5 Support Vector Regression (Machine)

Support Vector Regression (SVR) is a special type of Support Vector Machine. Support Vector Machines can be used to predict different classes, where the goal is to find a higher dimension which maximizes the distance between the different classes. Support Vector Regression is an adaptation used to give continuous predictions, instead of choosing between classes.

When trying to predict a continuous value instead of a class the goal becomes to find a tube which best contains the training points. This can be seen in Figure 10, the tube is of width ε while points outside the tube are denoted with ξ if the prediction is larger than the values of the tube, and denoted by ξ* if they are smaller than those in the tube. Due to the quadratic programming used to solve this problem these need to be represented by different variables. The minimization objective

for this tube consists of two parts: One part which minimizes the distance between the edge of the tube and the observations outside of it, and one part to minimize the weights of the tube. There are two parameters of the model, the first one is ε, which is the width of the tube. The second parameter U, is the multiplication factor for the error of distance of points outside the tube in the minimization objective. A low value of U will mean a high focus on generalization while a high value will mean that more focus is given to fitting the tube as close as possible (Drucker, Burges, Kaufman, Smola, & Vapnik, 1996).

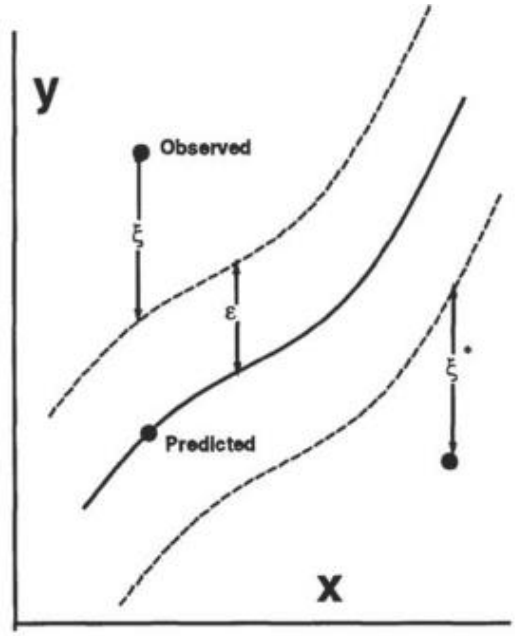


*Figure 10 - Support Vector Regression (from Drucker et al. p. 161)*

### 3.4.6 Model comparison

To finish off the literature chapter we will compare the strengths and weaknesses of the different models, and explain why we chose to include them in our research. Our final model we discussed in this chapter is Support Vector Regression, which is mainly included as a working model of this type would be extremely useful for our case, and give us a lot of insights. It is more unconventional however, and may end up not working well, although the fact that Gonsalves and Patil (2016) used this model gives reason to believe it may work.

One model type which we expect to work well are the Neural Network variants. The Neural Network model is used very often as it is extremely versatile while being able to find hidden patterns. There are two major downsides to the standard Neural Network. The first one is the risk of overfitting, which tends to become more of an issue when a model has more complexity. Large NNs have the potential to find a lot of interesting patterns, but the size gives them a larger risk of finding patterns which do not generalize. To combat this we have the Dropout Neural Network included in our study. Although it takes longer to train it may work better when the model contains more nodes and/or layers. The second big problem of NNs is the required training time. As expected, the more layers and nodes the more calculations are needed to update each node. If this becomes an issue we

can use the Sparse Neural Network, which drops nodes to reduce the necessary calculations. Our final two models are less complex than the previous ones.

The K-nn has a very basic premise of trying to find the most similar datapoint from historical records to predict new data. This means that all of the previous datapoints need to be compared to the new one which means that prediction time increases exponentially with the amount of training data points. At the same time, if there are few historic datapoints the model may have a worse than average time in predicting new datapoints. However, due to the number of datapoints in our training set being neither too few nor too many, it is appropriate in this case. And since Lin et al. (2019) used a K-nn variant as part of their model to predict sparse input data, we choose to include this model as we have the same problem with our data.

Finally we also include Gradient Tree Boosting. This model is more often used for predicting classes but can be used for regression problems as well. We include this model as training time is not as long as most Neural Networks, while being strong at finding certain patterns. If it happens to be the case that these patterns are found often in the data this model becomes very useful. We have a lot of categorical data which the Gradient Tree Boosting can use better than continuous data, which it would need to split up manually. If this model ends up working well it is the most transparent model in our selection, and would yield many insights into which projects do well and which do not. The potential, even it is not the most likely candidate to succeed, is why we decided to include this model.

# 4. Practical methodology.

After having discussed the literature side of our methodology in the previous chapter, we will explain our adjustments and new additions. We start in section 4.1 by looking at the possibilities and limitations of the predictions our models can make. From this we find that scaling up the size of Neural Networks keeps marginally improving performance, at ever greater computational expense. To limit the time training takes we develop a Genetic Algorithm in section 4.2 which limits the product of the two greatest contributing factors to this training time: the number of layers and the number of nodes. The algorithm looks at the combined value instead of looking at just the singular parameter value a regular Genetic Algorithm would look at. Finally we develop a backwards prediction analysis discussed in section 4.3 as we want to validate the data our models uses, and have a check to see if we can improve the prediction accuracy by reducing the number of poor features. This is done as we have found in section 4.1 that our models struggle to be more accurate than the difference between the budgets of projects and their actual spending.

## 4.1 Model exploration.

This section discusses several observations, problems found, and decision made the search for a model to predict project spending. The first topic discussed is what we are predicting and what data we are using for this. This is followed by a section on how we deal with overfitting. The final section discusses the removal and reason for removal of a single feature, which is identified by a separate analysis. At the end of this section 5 tables with the results of some of our experiments can be found, which show some of the problems and success our models have.

### 4.1.1 What can be predicted with the data?

During most explorations to find good models the smaller dataset is used. This choice is made because there were a good number of projects available for each of the four countries (Ethiopia, Egypt, Nigeria, Vietnam) while keeping the dataset small enough to limit the time needed to train the models. The countries are varied in cultures and project profiles. The input for all models consists of data on the project level. For each project the budgeted amount is available as well as a number of project descriptors such as the country the project is taking place in and the target age group. We look into the possibility of predictions made for the Field spending per project per year. Here models such as K-Nearest Neighbor, Neural Networks and Support Vector Machines are tested. These either try to predict the total amount of money which is going to be spend, or they try to predict the percentage of the planned budget which is going to be spend on the projects, two examples can be seen in Table 2 and Table 3. For both types it is tested if using the output from a K-nn into an NN could increase performance, which is not the case, one of the results can be seen in Table 1. Here it also shows that performance of percentage models is considerably worse that those which directly predict the expected spending. The best results for the small data set predicting 2016-2018 prove it is very possible, and some of the NNs have an Mean Absolute Deviation (MAE) which was 25% less than the MAE of the budget compared to the actual spending, as seen in Table 4. As mentioned before the MAE is the average difference between a prediction and the actual value, where being -100 USD or +100 USD are both seen as being wrong by the value of 100 USD. The parameters of the models are varied widely to find good models, and for the Neural Networks several optimizers and activation functions are tested, along with varying the width and depth of the network.

We also attempt to predict monthly expenses per project. From our analysis in chapter 2.4 we find that having a detailed prediction would be more beneficial for the NGO than yearly predictions. The reason is that this would help them with cashflow planning as well as help in keeping track of how well projects are going. After extensive model exploration it becomes apparent that predicting each month separately would not be feasible. Some of our best models have an average MAE per month of 85%, with no months being of by less than 50%. As this is too far off any useful predictions the choice is made to attempt to make quarterly predictions instead. Along with this we also remove some big outliers. Project which had (unscaled) underspending or overspending of 100.000 USD or more are removed from the dataset, as changes in spending this big needed to be approved by Management and were due to special circumstances. These special circumstances are generally outside the control of the NGO and small in number. Therefore the decision is made to remove these few datapoints as they do not fit the pattern we are trying to predict.

While making these monthly predictions we find that 2018 was considerably different from the previous two years. This is confirmed by the NGO and they expect the coming years to be more like 2018 than the previous years. The 2018 spending was much closer to the budget, and this means that for the 3-year dataset, the model is only better than the budget for the first two years. After splitting up the datasets it shows that models which are more accurate than 2016-2018 budgets, are not able to beat 2018 budget for the full dataset, not when only training on 2018 data, nor when training on all three years. The smaller dataset has models better than the 2018 budget, although not by much, two models predicting this dataset can be found in Table 5 and Table 6.

When predicting the quarterly spending we find that while quarterly predictions themselves are not accurate to the point that we considered them usable, adding them up does work to estimate yearly expenses. The previous models have one specific output for yearly expenses while the quarterly implicit NNs are able to perform nearly as well. We refer to these models as implicit as in contrast to our other yearly spending models they do not explicitly predict the yearly spending itself, but rather the underlying quarterly spending which leads to yearly spending. In order to confidently say that the quarterly models are equal to or better than the yearly models statistics should be used to compare the two performances. From a statistics point of view the differences are significant and worse for the quarterly models, however we do not see this as enough reason to use explicit models. Statistically each mean is different between the different models, which is mainly due to both datasets having well over 1500 datapoints. Such a large dataset has a very small interval in which the mean is considered statistically similar to others, which comes from the statistical formula used to compare two means. Concluding the performance of the models are different is fair as most models have different parameters. However results from running the same model twice are often also statistically different, the differences being caused by randomly ordering datapoints in the training. The best models which explicitly predict the yearly expense seem to perform marginally better than the implicit models, however for two reasons we decide to go forward with the implicit models. The first reason is that that the risk of overfitting is higher with the explicit model. While not the case in our tests, this is a risk for future models, and since the datasets are short in terms of time, we have concerns about the ability of the model to properly generalize. Secondly a quarterly model

better fits the needs found in chapter 2.4. So keeping in consideration possible future updates and applications of the model we choose to train implicit quarterly models, which when operational would better fit the needs of the NGO.

Along with our successful models predicting yearly expenses through implicit models we also investigate making separate models for each different quarter. Unfortunately these models do not perform better than combined models and overfit very badly despite our measures to combat this. The overfitting once again manifested itself through performing extremely poor on one or two folds from our 5-fold testing, an example of which can be seen in Table 6. Many NN configurations are tested but eventually due to the extra computations, overfitting and worse performance we decide not to include this setup for model optimization. During our testing we repeatedly have very poor performance from the Support Vector Regression model, and we decide to focus our attention on more promising models. Therefore we do not consider this model in the future sections of our research. From our many experiments we come to the conclusion that the best prediction fit for our problem is to make quarterly predictions, and take the sum of these predictions as yearly expense predictions.

### 4.1.2 Preventing models from overfitting

During the previously mentioned exploration of prediction level we have severe overfitting, in particular when we train deep NNs, some of which have good very performance. Our solution to this is to use Dropout and Sparse NNs which prevent the overfitting. From the many possible parameter configurations we test, most of the deeper models do not work well. From the models which performed better we find them to be prone to significant overfitting. This shows by often having one fold, and sometimes two folds of the 5-fold validation sets strongly overfit causing an average very poor performance. As a reminder, this can be seen in table Table 6. The splits which do not overfit however have the best performance of any model we tested, which is why we investigate how to combat the overfitting of our deep models. Deeper NNs in general have an increased risk of overfitting which is true for our models as well.

We test several ways to reduce overfitting: Lasso, Ridge and Dropout, as well as Sparse networks. All methods get rid of the extreme overfitting, but both Dropout and Sparse NNs prove to be positive additions for the predictions. This allows us to make larger models without them overfitting, however the best models require too much processing power to be properly explored. For all deeper NNs we only use the ReLu activation function, as this is the common activation function for deep NNs. We do try models using ELU and Leaky ReLu, but these did not work well, hence the choice to use only ReLu activation. The fact that ReLu worked better than ELU and Leaky ReLu leads us to believe that it might help against overfitting. The main problem these alternatives to ReLu address is the prevention of nodes dying off. When this happens the node is no longer able to update due to the output value always being less than 0, which due to the structure of the ReLu activation function means that the nodes always outputs 0. This causes the derivative to be 0 at this node at all times meaning it cannot learn new weights anymore. As the node can no longer function it is considered dead. Although not tested we theorize that the dying of the ReLu nodes have a better performance than non-dying variants due to the dying of nodes combatting overfitting. We

think it might have a similar effect as Dropout, although it is not explicitly looking for the best nodes to prune.

### 4.1.3 Feature removal

Our last finding is limited in importance to the scientific side of this thesis, but is valuable to the NGO. The dataset consists of some features which are expressed in dollars, and all but one of the other features are categorical. This one other continuous feature, referred to as feature N is not expressed in dollars, and turns out to be a very poorly recorded feature. When treating feature N as a categorical feature and one-hot encoding it, that is, making a new column for each value of the feature, the performance is similar to not including it. Including it as a continuous feature results in a drop of over 500 USD of the MAE for both the 2016-2017 and 2018 datasets for multiple of our best models at this time. This is for a dataset for 4 countries over 3 years which indicates that the consistency and/or accuracy of feature N is poor. After discussing this with the NGO there were already some concerns as to the way feature N was recorded, although they did not expect it to be this poor. Our advice would be to revise this feature and split it up. Currently part of the perceived problem is ambiguity in what is meant by N. This seems warranted as there are multiple fair interpretations for different kinds of projects. We think that splitting it so each of these different interpretations are different features would result in better records as well as provide useful measuring data. Having a better idea of the projects want to achieve and measuring the results afterwards gives a better picture as to how well each project performs and would help in identifying good projects. For the model however feature N will be excluded from the dataset.

**Small dataset, K-NN as input for NN model - Dollars spent as prediction goal**

| Agregated monthly predictions | **Prediction bias** | **Prediction std dev** | **Prediction MAE** | **Budget bias** | **Budget std dev** | **Budget MAE** | **Epochs** |
|---|---|---|---|---|---|---|---|
| **Average** | -5,312858 | 11,66068 | 6,507508 | -1,08956 | 4,531061 | 1,920708 | 30,8 |
| | | | | | | | |
| **fold 1** | -5,773371 | 9,6704 | 5,846588 | -1,69739 | 4,40583 | 1,956359 | 57 |
| **2** | -5,740962 | 7,967604 | 5,885444 | -0,87421 | 3,024085 | 1,543583 | 61 |
| **3** | -4,213165 | 14,05681 | 6,645589 | -1,37813 | 5,26642 | 2,139036 | 15 |
| **4** | -3,012489 | 12,78143 | 6,335616 | -1,14441 | 3,628378 | 1,714004 | 19 |
| **5** | -7,8243 | 13,82716 | 7,8243 | -0,35364 | 6,330593 | 2,250558 | 2 |

*Table 1 - Small dataset, K-nn predictions as input for a Neural Network model - Dollars spend as prediction goal*

**Small dataset, Basic NN - % of budget spend as prediction goal**

| Agregated monthly predictions | Prediction bias | Prediction std dev | Prediction MAE | Budget bias | Budget std dev | Budget MAE | Epochs |
|---|---|---|---|---|---|---|---|
| **Average** | 5,918287 | 9,870344 | 5,918287 | -0,95053 | 4,77096 | 1,79931 | 18,2 |
| | | | | | | | |
| **fold 1** | 5,432447 | 7,686345 | 5,432447 | -0,73552 | 3,197395 | 1,494944 | 7 |
| **2** | 7,093126 | 12,55154 | 7,093126 | -0,74902 | 6,722233 | 2,271946 | 24 |
| **3** | 5,367134 | 8,541766 | 5,367134 | -0,80024 | 3,801445 | 1,745397 | 7 |
| **4** | 5,210914 | 10,58867 | 5,210914 | -1,6474 | 6,5838 | 1,900286 | 21 |
| **5** | 6,487813 | 9,983405 | 6,487813 | -0,82049 | 3,549928 | 1,583978 | 32 |

*Table 2 - Small dataset, Neural Network - prediction percentage of budget spend as prediction goal*

**Small data set, Support Vector Regression - % of budget spend as prediction goal**

| Agregated monthly predictions | Prediction bias | Prediction std dev | Prediction MAE | Budget bias | Budget std dev | Budget MAE |
|---|---|---|---|---|---|---|
| **Average** | 5,456132 | 9,937602 | 5,49427 | -0,95011 | 4,777814 | 1,799767 |
| fold | | | | | | |
| **1** | 6,293964 | 10,89243 | 6,347255 | -0,57658 | 3,673659 | 1,425644 |
| **2** | 4,749185 | 11,50318 | 4,779512 | -1,25342 | 6,968609 | 1,971368 |
| **3** | 5,694832 | 9,903464 | 5,729417 | -1,22205 | 4,440082 | 2,110793 |
| **4** | 5,692327 | 10,40107 | 5,724548 | -0,5124 | 6,027808 | 1,908263 |
| **5** | 4,850352 | 6,987868 | 4,890619 | -1,1861 | 2,778912 | 1,582765 |

*Table 3 - Small dataset, Support Vector Regression - Percentage of budget spend as prediction goal*

**Small dataset, Sparse Neural Network - Dollars spend as prediction goal**

| Agregated monthly predictions | Prediction bias | Prediction std dev | Prediction MAE | Budget bias | Budget std dev | Budget MAE | Epochs |
|---|---|---|---|---|---|---|---|
| **Average** | 0,211129 | 3,843257 | 1,515272 | -0,95029 | 4,933424 | 1,800035 | 19,8 |
| | | | | | | | |
| **fold 1** | 0,653944 | 5,330359 | 1,418031 | -0,47202 | 5,351411 | 1,554232 | 17 |
| **2** | -0,804379 | 2,655206 | 1,269336 | -1,21126 | 6,45654 | 1,625847 | 3 |
| **3** | 0,431385 | 3,546214 | 1,87724 | -1,17519 | 4,337906 | 2,199967 | 19 |
| **4** | 0,313616 | 4,697582 | 1,503649 | -0,74268 | 4,735666 | 1,886532 | 19 |
| **5** | 0,461079 | 2,986921 | 1,508102 | -1,1503 | 3,785599 | 1,733595 | 41 |

*Table 4 - Small dataset, Sparse Neural Network - Dollars spend as prediction goal*

**2018 Small dataset, Sparse Neural Network - Dollars spend as prediction goal**

| Agregated quarterly predictions | Prediction bias | Prediction std dev | Prediction MAE | Budget bias | Budget std dev | Budget MAE | Epochs |
|---|---|---|---|---|---|---|---|
| **Average** | 0,15154 | 1,192728 | 0,719446 | -0,35554 | 1,059375 | 0,591448 | 15,6 |
| | | | | | | | |
| **fold 1** | -0,118142 | 1,504354 | 0,760426 | -0,38557 | 1,440518 | 0,755793 | 11 |
| **2** | 0,286027 | 1,159726 | 0,763047 | -0,5311 | 1,190005 | 0,660079 | 27 |
| **3** | 0,375809 | 1,301137 | 0,797659 | -0,25552 | 0,869828 | 0,510435 | 2 |
| **4** | 0,050936 | 0,839258 | 0,544659 | -0,25158 | 0,704607 | 0,40313 | 32 |
| **5** | 0,163072 | 1,159167 | 0,73144 | -0,35391 | 1,091915 | 0,627805 | 6 |

*Table 5 - Small dataset with only 2018 data, Sparse Neural Network - Dollars spend as prediction goal*

**Overfit 2018 Small dataset, Sparse Neural Network - Dollars spend as prediction goal**

| Agregated quarterly predictions | Prediction bias | Prediction std dev | Prediction MAE | Budget bias | Budget std dev | Budget MAE | Epochs |
|---|---|---|---|---|---|---|---|
| **Average** | 0,405992 | 1,665409 | 0,983067 | -0,35564 | 1,084281 | 0,591302 | 15 |
| | | | | | | | |
| **fold 1** | 0,185768 | 1,278396 | 0,782717 | -0,42429 | 0,935219 | 0,550969 | 1 |
| **2** | -0,133986 | 1,36079 | 0,769208 | -0,40738 | 1,225605 | 0,68892 | 59 |
| **3** | 0,4313 | 1,486302 | 0,961628 | -0,21282 | 1,225734 | 0,557065 | 9 |
| **4** | 1,594574 | 3,198196 | 1,76839 | -0,34079 | 1,075564 | 0,588796 | 3 |
| **5** | -0,047695 | 1,00336 | 0,633392 | -0,3929 | 0,959284 | 0,57076 | 3 |

*Table 6 - Small dataset with only 2018 data, overfit on fold 4, Sparse Neural Network - Dollars spend as prediction goal*

## 4.2 Adjusted Genetic Algorithm

In chapter 3.3.5 we discussed the Genetic Algorithm of Leung et. al. (2003). In our application of this algorithm some adjustments are made to accommodate for the processing power available. From testing it shows that, while the relation is not linear, the main driver for training time is the total number of nodes in the entire network. This is a product of the number of layers and the nodes per layer. Therefore a maximum is set on the combination of the number of layers and the number of hidden nodes per layer. For the GA the number of nodes per layer are calculated by taking 2 to the power of $n$, and the number of layers was $m$. The combination of $n + m < maxSize$. If this constraint is violated when generating the first four offspring the following algorithm is used to reduce the number of nodes and layers.

Here $n_{nodes}$ refers to the dynamic variable of number of nodes, which changes during the operations, while $pref_{nodes}$ refers to the number of nodes the original genetic algorithm tried to assign. Our adjusted algorithm tries to keep some balance between the two values. For the four

crossover operations if the parameters start equal, they will end up (nearly) equal, and if there is a large difference the algorithm will attempt to keep the same proportions: 5 layers and 10 nodes will become 4 layers and 8 nodes, which both have proportions of 1:2. The algorithm will also prevent one of the values from becoming 0 which would lead to invalid models. There is a small bias towards reducing nodes, as from experiments with the smaller data set it became clear that deep networks perform significantly better than shallow ones, while widening has an improvement which is comparatively smaller.

The adjustments made for the random mutations are less sophisticated and there could be better algorithms, but that falls outside the scope of this project. The method used for the first two variants, one with just one change, the second with random chance for each parameter to chance, is to generate a random value in the interval $[1 : (maxSize - \ the\ other\ parameter)]$. For the third mutation every parameter is changed in the original algorithm. Randomly, with a 50% split, either nodes or layers is changed first, within a maximum value of $maxSize - 2$. This ensures the other parameter will always be able to change. While this prevents 11-1 type splits these are still possible for the other two mutation methods, and since a few hidden nodes with 11 layers will not create a good model, and a deep network worked performed considerably better in test this small restriction seemed acceptable. After the first parameter is set, the other parameter is randomly chosen in the interval $[1 : (maxSize - \ the\ first\ parameter)]$. For the standard NN the applied algorithm can be found in Figure 11.

Our algorithm allows the minimum and maximum number of layers and nodes can be in the interval $[1, maxSize\ - 1]$. Without our adjustments the interval would be $[1, maxSize/2]$ or similar. This would mean that NNs with 8 layers with few nodes, or 3 layers with many nodes are not possible. Our algorithm is useful when there is a constraint involving multiple parameters, for example training time, which need to be optimized.

$$
\begin{aligned}
&If\ \ n_{nodes} + n_{layers} > 12\\
&\quad While\ \ n_{nodes} + n_{layers} > 12\\
&\qquad If \frac{n_{layers} - 1}{pref_{layers}} > \frac{n_{nodes} - 1}{pref_{nodes}}\\
&\qquad\quad n_{layers} = n_{layers} - 1\\
&\qquad Else\\
&\qquad\quad n_{nodes} = n_{nodes} - 1
\end{aligned}
$$

*Figure 11 - Algorithm to reduce the nodes and layers parameters*

## 4.3 Backwards prediction analysis

For our research we analyze the input variables in some kind of backwards analysis. This is an analysis which is novel in its use, but based on principles from literature. Instead of looking "forward" at the data we want to predict, we look "backwards" at the data which is supposed to predict the new data. For our backwards prediction we see if we can predict our input data. As we find that our models are not able to perform as well as we are aiming for, we want to know if this is due to the information coming in being inadequate. We need to validate the data and see if there are relations between the different variables. Since the prediction goal is used to check for correlation if

there are barely any or no relationships these features may be poorly recorded or unrelated to the project spending, in which case they will likely hinder our models from learning how to predict properly. With the intent of the thesis being to help the NGO better predict Field spending, we want to do something different than a PCA as we would like to give insights into their data's ability to predict different features and categories of these features. Taking inspiration from the backwards predictions we consider that if (part of) the input data cannot be predicted, it will mean that forward predictions are very likely to be of poor quality, as mentioned in chapter 3.3.2. Our backwards prediction takes all data, both the data which would normally be input data, as well as the data which we are trying to predict and merges this into one dataset. One at a time a feature is removed and a model tries to predict the value of this feature using the remaining features. The advantage of this backwards prediction is that with the categorical variables it is possible to see which categories are easy to predict and which are harder. This allows for more targeted advice compared to only knowing that a feature is not working well. Furthermore it differentiates between a feature where every category is roughly equal in accuracy and between features where certain categories perform much better than others. This increased level of detail makes it more interesting when looking for long term improvement as opposed to an analysis such as a PCA which is purely focused on immediate gain, but yield little to no insights as to which features work well.

## 4.4 Methodology conclusion

From our model exploration we find that the most suitable prediction objective is to predict the quarterly spending and sum this to find an estimate for the yearly expenses. To combat overfitting for the NNs we use Sparse and Dropout NN when the layers and nodes per layers become large. We also exclude a feature recorded for the project statistics as it is not recorded well enough. To find good model parameters while keeping computational times reasonable we design a Genetic Algorithm which uses a maximum for two parameters combined, in our case the number of layers in a NN and the number of nodes per layer. This algorithm reduces the parameters in a balanced way if they exceed an allowable total. Finally we design a backwards analysis to check for data validation and for feature selection.

# 5. Results

In this chapter we will discuss the results of the models and analysis we found in the previous chapter. We start with our backwards prediction analysis and then the results of our final models are discussed.

## 5.1 Backwards prediction analysis

As part of us analyzing the predictability of the final spending, a backwards prediction has been done. This is done on the smaller dataset, as well as the entire dataset. The main goal is to find out whether there are completely uncorrelated or poorly recorded features, and to see if the categorical features have certain categories which perform much worse than the others. This will allow us to give better advice to the NGO with regards to gathering data in the future to improve on the performance of the predictive models. If any feature does very poor we can also remove this feature from our dataset. For this prediction a neural network is trained on all of the training data except for 1 categorical feature, which it tried to predict. This NN has one 512 node hidden layer with ReLu activation, the final layer is sigmoidal. The updates are done in batches of 16, the optimizer is RMSprop and MSE is the minimization objective. We experiment with other minimization objectives as well, but the result of our first different objective Mean Absolute Error outputs predictions with values of 0. Having the Mean Absolute Error as minimization goals means the model is unable to train. MAPE is able to make some distinctions between categories but the performance of the predictions is poor as well. The results from the smaller dataset are mixed, as certain categorical features proved to be much easier to predict than others.

Each model is trained with a stratified K-Fold where K=5. The model parameters are not fully optimized but given previous experiments the parameters are expected to be adequate in case of strong predictability. If the features can be predicted with good accuracy this indicates a well-defined category and the presence of predictive power from the feature. Low predictability indicates the opposite, and raises questions about the accuracy of the data, whether the categories of a feature are sufficiently different from another and whether the datapoints in the category might be heterogeneous. The outcome of our main analysis can be found in Table 7. In general performance of the model is expected to decrease when the number of different categories increase, as the number of training samples for each category decreases. The models would output a value between 0 and 1 for each different category. The category with the largest value is considered the predicted category.

| Feature | Egypt, Ethiopia, Nigeria, Vietnam | | All countries | |
|---|---|---|---|---|
| | Number Of categories | Percentage correct | Number Of categories | Percentage correct |
| *Country* | 4 | 96.8% | 59 | 70.9% |
| V2 | 9 | 81% | 13 | 73.8% |
| V3 | 7 | 61.3% | 7 | 66.1% |
| V4 | 5 | 68.6% | 5 | 76.6% |
| V5 | 13 | 40.9% | 10 | 56.1% |

| | | | | |
|---|---|---|---|---|
| V6 | 34 | 52.4% | 34 | 47.1% |
| V7 | 6 | 81.5% | 7 | 74.9% |
| V8 | 4 | 64% | 5 | 74.5% |
| V9 | 3 | 69.4% | 4 | 78.7% |
| V10 | 3 | 86.8% | 3 | 92.6% |
| V11 | 8 | 78.4% | 8 | 73.1% |
| V12 | 7 | 78% | 8 | 84.2% |
| V13 | 7 | 46.2% | 5 | 66.1% |
| V14 | 4 | 88.8% | 4 | 89,9% |
| V15 | 3 | 91.3% | 3 | 88.4% |
| V16 | 3 | 91.7% | 3 | 89.7% |

*Table 7 - Prediction accuracy of the backwards prediction by a 512 node Neural Network. Notable are the inconsistency between performance and number of categories, showing some are more consistent and corelated than others*

Notable outcomes are that the country and the V15 and V16 features are very predictable while V5, V6 and V13 all scored very low. V5 included 4 categories with less than 10 data points and another with less than 20. V13 has similar problems as 5 out of the 7 categories covered less than 4 percent of the datapoints, yet despite the low number of training samples for these tiny categories a large number of guesses are for these categories, showing a poor distinction between them. These results show some indications of overlap, although some of the false predictions are likely due to too few training samples, in particular those from very small categories.

We notice a clear gap between the worst three features, which are worse by 7.3%, 22.4% and 27.7%, compared to the next worst feature. We are interested to see if this is due to the Neural Network parameters being poor or if they are really worse in quality. More experimentation is done to see if there are some harder to find connections and these show improvements. The V13 feature improved to 89.2% accuracy when expanding the model to have 3 layers of 512 nodes. Other configurations tested are just adding a second layer of 512 which proved similar in performance to widening a one-layer model to have 4096 nodes, with both having just under 61% accuracy. Every model improved and our best model performed very well, showing that V13 has a more complex relationship with other features, but clearly has some underlying relations.

The feature V3 is best predicted by a one layer 4096 nodes model and went to 60.5% accuracy. Other alternatives tested are a 3 layer with 512 nodes each model, 4 layers of 512 nodes each and single layer 8192 nodes models, all performing worse. The V5 feature did not improve, the best one just 44% accuracy with the 3 layers 512 nodes each model. For 2 of the features at least part of the problem appears to be too few training samples. Therefore a new training run is done for V13 and V5, where any category with less than 10 training samples has their datapoints removed, to see if this will allow the category prediction model to improve in accuracy. For the V13 feature this did not improve with the model consisting of 3 hidden layers of 512 nodes scored 80.8%, almost 9% worse than when the extra categories are included. For V5 the best model has 4096 hidden nodes with an accuracy of 44.6% barely better than when the small categories are included.

The results indicate that most of the features do contain a considerable amount of distinctive information, although the features V5, V3 and V6 should be given extra attention from the NGO to improve data collection. V6 has a large number of categories, which appear to be too many, both for the model, as well as for several other practical applications. While the feature country cannot be reduced further this is not the case for V6. For the NGO we would advise to look into the V5 feature, to see if the different categories are used correctly, if they are really different from each other and whether the categories might contain significantly different categories. The need for a relatively complex model to predict the V3 is also a red flag that we believe could be worth looking into.

## 5.2 Final models

As we have found our models and the way we can optimize them we trained the different models and analyzed the results. The outcome of training our models to predict yearly project expenses can be found in Table 8. For the yearly predictions we can look at the difference between the budgeted amounts and the actual spending. Our best model, the K-nn, is off by an average absolute difference of $1.901 per project compared to the budget being off $1.866 per project for the 2016-2017 period. For the 2018 period the K-nn was also our best model, being off $1.241 (121%) per project while the budget was off by $1.023 (100%), the second and third best being the Boosted tree with $1351 (132%) and the Dropout NN with $1491 (146%), when comparing MAE. Since the budget information per quarter is not available we cannot compare this to see how much better or worse the predictions are in comparison to the actual spending. However when comparing the error of the individual quarters and the combined yearly prediction it becomes clear that quarterly predictions are not yet feasible. The quarterly results can be found in Table 9 for Q1, Table 10 for Q2, Table 11 for Q3 and Table 12 for Q4. The amounts in all tables are scaled amounts in thousands of USD. In 2018 the completed average project would report the spending in each successive quarter respectively: 39%, 14%, 22% and 24% in Q4. For the 2017 and 2016 dataset this is 47%, 13%, 18% and 22%. This pattern is similar to the pattern of the Standard deviation as well as the Mean Absolute Error.

When looking at the bias the only model which is considerably off the 0 dollar goal is the 2016-2017 Sparse Constant NN. This is not unsurprising given the complexity of the model, although it being worse than the Standard NN is an interesting result. Also interesting is that the bias for the Budget compared to Actual for 2018 in Table 8 has a worse bias compared to all models, and is the second worst bias out of all final experiments. The NGO was concerned about the Field underspending on their budgets, which gets confirmed for the entire period and is really considerable for 2018. Where Lin et. al. (2019) used a hybrid model including a K-nn component for their sparse data, this also works well for our data. The K-nn is also best in terms of the standard deviation of the error, and while not the best the bias is still very acceptable. Therefore from the models tested we consider the K-nn model to be the best. The variant of this model we found to be best has weights based on distance. The datapoints of historical projects are weighted by the inverse of their distance to the predicted datapoint. Because the number of projects is not too high to result in too much processing power being needed, but high enough to differentiate between different projects, the good performance also makes sense from the theoretical point of view.

| Yearly Predictions | | Bias | Error Std dev | MAE |
|---|---|---|---|---|
| **Budget compared to Actual** | 2016-2017 | -0.117 | 3.888 | 1.866 |
| | 2018 | -0.328 | 2.374 | 1.023 |

| | | | | |
|---|---|---|---|---|
| **Standard NN** | 2016-2017 | -0.124 | 4.643 | 2.300 |
| | 2018 | 0.242 | 3.308 | 1.501 |
| **Dropout NN** | 2016-2017 | 0.198 | 4.584 | 2.225 |
| | 2018 | 0.077 | 3.207 | 1.491 |
| **Sparse Constant NN** | 2016-2017 | 0.807 | 4.706 | 2.366 |
| | 2018 | 0.244 | 3.440 | 1.549 |
| **K-Nearest Neighbours** | 2016-2017 | 0.035 | 3.990 | 1.901 |
| | 2018 | 0.138 | 3.301 | 1.241 |
| **Boosted Tree** | 2016-2017 | -0.185 | 5.304 | 2.317 |
| | 2018 | 0.021 | 3.371 | 1.351 |

*Table 8 – Statistics of yearly predictions of total spending, comparing historical results with our models. For 2018 the best models are in order K-nn, Boosted Tree and Dropout NN Amounts x $1000*

| **Q1** | 47%/39% | **Bias** | **Error Std dev** | **MAE** |
|---|---|---|---|---|
| **Standard NN** | 2016-2017 | 0.262 | 4.425 | 1.853 |
| | 2018 | -0.051 | 2.837 | 1.225 |
| **Dropout NN** | 2016-2017 | -0.078 | 4.742 | 1.866 |
| | 2018 | -0.102 | 2.809 | 1.243 |
| **Sparse Constant NN** | 2016-2017 | -0.374 | 4.219 | 1.736 |
| | 2018 | 0.038 | 2.926 | 1.278 |
| **K-Nearest Neighbours** | 2016-2017 | 0.020 | 4.644 | 1.816 |
| | 2018 | -0.053 | 2.861 | 1.137 |
| **Boosted Tree** | 2016-2017 | 0.022 | 4.887 | 1.838 |
| | 2018 | -0.023 | 2.945 | 1.175 |

*Table 9 - Prediction of spending in Q1, amounts x $1000*

| **Q2** | 13%/14% | **Bias** | **Error Std dev** | **MAE** |
|---|---|---|---|---|
| **Standard NN** | 2016-2017 | -0.043 | 2.911 | 0.813 |
| | 2018 | -0.051 | 1.767 | 0.680 |
| **Dropout NN** | 2016-2017 | -0.073 | 2.735 | 0.809 |
| | 2018 | 0.014 | 1.726 | 0.685 |
| **Sparse Constant NN** | 2016-2017 | -0.229 | 2.577 | 0.706 |
| | 2018 | -0.085 | 1.729 | 0.653 |
| **K-Nearest Neighbours** | 2016-2017 | 0.011 | 2.747 | 0.834 |
| | 2018 | -0.029 | 1.550 | 0.661 |
| **Boosted Tree** | 2016-2017 | 0.060 | 3.216 | 0.882 |
| | 2018 | 0.007 | 1.630 | 0.685 |

*Table 10 - Predictions of spending in Q2, amounts x $1000*

| **Q3** | 18%/22% | **Bias** | **Error Std dev** | **MAE** |
|---|---|---|---|---|
| **Standard NN** | 2016-2017 | -0.016 | 2.284 | 1.011 |
| | 2018 | -0.097 | 2.073 | 0.874 |
| **Dropout NN** | 2016-2017 | -0.058 | 2.277 | 1.005 |
| | 2018 | -0.003 | 2.021 | 0.917 |
| | 2016-2017 | -0.017 | 2.845 | 0.998 |

| | | | | |
|---|---|---|---|---|
| **Sparse Constant NN** | 2018 | -0.132 | 1.878 | 0.820 |
| **K-Nearest Neighbours** | 2016-2017 | -0.051 | 2.225 | 0.939 |
| | 2018 | -0.009 | 2.022 | 0.870 |
| **Boosted Tree** | 2016-2017 | 0.033 | 2.641 | 1.015 |
| | 2018 | 0.023 | 1.936 | 0.816 |

*Table 11 - Predictions of spending in Q3, amounts x $1000*

| Q4 | 22%/24% | Bias | Error Std dev | MAE |
|---|---|---|---|---|
| **Standard NN** | 2016-2017 | -0.080 | 3.943 | 1.281 |
| | 2018 | -0.043 | 1.789 | 0.801 |
| **Dropout NN** | 2016-2017 | 0.012 | 4.570 | 1.391 |
| | 2018 | 0.014 | 1.723 | 0.788 |
| **Sparse Constant NN** | 2016-2017 | -0.186 | 3.696 | 1.187 |
| | 2018 | -0.066 | 1.847 | 0.777 |
| **K-Nearest Neighbours** | 2016-2017 | -0.014 | 3.461 | 1.223 |
| | 2018 | -0.046 | 2.210 | 0.867 |
| **Boosted Tree** | 2016-2017 | 0.070 | 4.624 | 1.389 |
| | 2018 | -0.029 | 1.917 | 0.826 |

*Table 12 - Predictions of spending in Q4, amounts x $1000*

## 5.3 Results conclusion

From our results we have two main findings. From our first section we find that none of the features are uncorrelated with the rest of our data set, and that some of them are strongly correlated. The accuracy of the Country feature is high, given the 59 countries included in our large data set and the 70.9% accuracy. This indicates that each country has some distinct combinations of projects which makes them unique from other countries. We did find that some of the features, in particular V3, V5 and V6, could use some attention from the NGO as V3 requires a complex model to explain well while not being a complex feature. V5 and V6 both having a relatively poor performance, with V6 also having a large number of categories, and we question if the large amount is helpful. For V5 we are also unsure whether the categories are used correctly. Overall though it appears like every category of the dataset is of sufficient quality to use in our model.

In our final models we test six different models by using a Genetic Algorithm to find good parameters. From this we find that the K-Nearest Neighbors model is the best of our models with a very low bias, the best Mean Absolute Error for both time periods and a very good average standard deviation of the error. For the entire data set none of the models are able to be more accurate than the budgets. The quarterly predictions which are the underlying parts used to make the yearly predictions are very poor in terms of MEA and average standard deviation of the error, but the bias is good for most models, with some exceptions of the Standard Neural Network and the Sparse Neural Network.

# 6. Conclusions and recommendations

Our conclusions are separated into three parts. First we revisit the original research questions and answer them in 6.1. Following these conclusions and our observations during the making of the thesis we write some recommendations to the NGO on how to move forward with regards to predicating final spending of their Field projects. After this we write a discussion and end with the limitations of our research.

## 6.1. Answering the research questions

With the results from chapter 5 we can answer our research questions from chapter 1.3. The first question we asked was:

1. What kind of predictions does the NGO need?
    - Which departments benefit from predictions?
    - What specific type of prediction would be most useful?

In chapter 2 we looked at how the NGO operates and what predictions could do for the different departments of the NGO. In Section 2.4 we find that for Management better predictions would help in making choices which projects to approve as there is more certainty of the expenses. For Field it would make managing the projects easier as they understand the spending pattern better. Finance would be helped through better cashflow planning which would be possible with better predictions. The biggest gains would be for Development who would be able to raise money with more restrictions as they have a higher certainty that it will be spent, and have a few instances of overfunding. Finally it will benefit the entire NGO as end of year reporting becomes easier since plans are closer to the actual spending. In Section 2.7 we find three types of predictions which may benefit the NGO:

1. Better predictions at the start of the year
2. Better predictions throughout the year
3. An understanding of the spending pattern throughout the year.

As the available data does not contain the necessary features to make updated predictions as new information comes in throughout the year, we try to make predictions of the first and third types.

Based on the needs of the NGO we look at the way we can meet these needs, and the research question we have for this is:

2. What models are likely to work?
    - What models have been used for similar problems?
    - How can common Machine Learning problems be prevented?

When looking at literature in chapter 3 we find that the research done on charities and NGOs is limited compared to some traditional industries. The main subject of the research is the psychology of giving money to charities. We find some related works when looking at healthcare, and find that for sparse data a K-Nearest Neighbors part is used, as well as some other models we use for benchmarking. We find that K-nn and Neural Networks are most promising, while also looking at Gradient Tree Boosting and Support Vector Regression. In chapter 2.3.2 we discuss over- and underfitting, one of the main problems with our dataset. Some of the methods we find are Sparse

NNs as well as Dropout NN, chapter 3.4.3, when the standard NN, chapter 3.4.2, would overfit. We look at heteroskedasticity in chapter 3.2.5, which occurs when there are multiple variances within a population. This is can be due to certain factors not being represented in the features used to predict the data.

To translate the theoretical models from the previous questions to practice we want to answer our final research question so we design our models:

3. How can we use the models from literature to predict spending?
    - How can appropriate parameters be found?
    - What can be predicted with the models and data available?

In chapter 4 we look at the practical part of finding a suitable model for our core problem of prediction project spending for Field projects. In 4.1 we conclude that it is not feasible to predict monthly or quarterly spending, the third type of predictions which would benefit the NGO. The MAE per month is on average at least 50% of the monthly average spending, and on average the best models would have an MAE of more than 85% of the budget. The first type of prediction does approach the accuracy of the budget, both while trying to predict quarterly spending and while predicting it directly. As the indirect method is less prone to overfitting, we choose to set this as prediction goal for our models. To deal with the processing time caused by both the number of nodes per layer and the number of layers for the NN we develop an adjusted Genetic Algorithm in chapter 4.2. This algorithm limits the total number of nodes which allows for models which do not reach the maximum combined size and for models which are very imbalanced while favoring using the all available processing power.

From our final results we find that the K-Nearest Neighbors model is most suited for our problem and dataset. However, this model is still not as accurate as the budget while still requiring project details including the planned size. The K-nn is off $1.241 (121%) on average per project while the budget is off by $1.023 (100%), the second and third best being the Boosted tree with $1351 (132%) and the Dropout NN with $1491 (146%), when comparing MAE. When more computational power becomes available Neural Networks will outperform the K-nn, which we saw occurring during model exploration. The main challenges in making the predictive model are heteroskedasticity, the overfitting of more complex model, as well as the data sparsity. From the large number of categorical variables, some with many different types, the resulting input vector contains many 0 entries. As yearly predictions are somewhat close to the difference between budget and actual spending, the same is not true for the underlying quarterly predictions. We have used an adjusted genetic algorithm to find good parameters, which performed well.

## 6.2 Recommendations for the NGO

Throughout working for the NGO and researching for this thesis we have gained insights into some of the operations and data gathering of the NGO. This leads to some advice on different areas of the NGO. Some of the advice is sensitive and will be part of an unpublished attachment, while the advice directly related to the research question will be discussed in this section.

The performance is not good enough to use for actual predictions to be able to support the LuKE system. Given the heteroskedasticity even in the models most adjusted to prevent this, it becomes clear that at least one unknown important factor is not included in the dataset of this study, and

there may be multiple unknown important factors. Unfortunately the model does not indicate what factor(s) is/are missing. From the difficulties from predicting monthly and to a lesser extent quarterly expenses at least one factor appears to be related to events/patterns which happen throughout the year.

Since there are multiple organizational changes in the structure and operations which have happened, are happening, and are planned to happen, it is advised to look into predictions again when the new work setting is relatively stable. As the basis for predictions is that the past is similar to the future it does not make sense to spend more effort on better predictions for the immediate future. One of the changes in the NGO is to do more extensive data gathering, which may find (some of) the missing data. If resources are available the Field teams could be approached or investigated to see if this can uncover factors which influence the spending patterns. In order to have sufficient data to train on we would recommend gathering several years of data for the new standard of working before trying to make a new model. While having a larger data set of for example 5 years of data would be preferable, given the dynamic nature of the operations this may not be realistic. If a new model is desired it will be a judgement call where the variance in Field operations and the time frame of the data will need to be weighed against the desired accuracy. It is also important to keep in mind that for a factor to be able to be taken into consideration it needs to be measured/available for each year of the historical data, as well as each future year the model will be used for. If a feature is no longer being measured the model will need to be retrained which can impact the accuracy of the model. While we would advise to run the all the models we have used and which are provided to the NGO we would expect that with more and better data, and possibly more processing power, the Neural Networks will be the best performing models. However at least the K-Nearest Neighbor should be tested as well, given the performance on the dataset used in this thesis.

While the historical data was from a stable and pre-corona workflow this is not the case for the currently gathered data, which makes it hard to impossible for a model to use it. As consistent data is not available and instability due both to corona as well as changes in workflow is not allowing for the gathering of this new high quality data we recommend a different approach. Since a higher accuracy is desired to enable full use of the new LuKE system, which matches projects and dedicated funds, our advice is to consider the most constraining project types and look to get a quick report from these projects regarding their planned and actual spending periodically. If the people who are in charge of actually spending the money are giving updates they should be somewhat accurate if the person knows how to budget. This person will also have a better understanding of reoccurring events which are not on the same date every year but occur with some regularity, such as the monsoon or festivals such as the Ramadan. If possible these persons may also provide insights into missing features for our model. Since most people do not enjoy giving such reporting it is important that the person understands the necessity and that the report itself is as short, quick and easy to use as possible. This will increase the chance that it is filled in correctly and with care, and prevents the people filling it in from being distracted by (perceived) bureaucracy.

## 6.3 Discussion

From our experiments we find that the K-Nearest Neighbors model is most suited for our problem. Considering the findings of among others Lin et. al. (2019) and Gonsalves and Patil (2016), who both use a K-nn variant in their models when working with sparse data, we think that the interactions between sparse data and K-nns could be used to develop better models. Researching how K-nn

may help predictions where there are gaps seems promising to us. There are different ways of using the K-nn as part of a larger model, be it a scalar, as dimensionality reduction method or as additional information to help models find new patterns. We performed some minor unsuccessful experiments in combining K-nn with a Neural Network, but due to overfitting and lack of good predictions we focused our attention on trying different models and combating the overfitting. We are interested both in the practical experimental possibilities as well as more theoretical mathematical approaches regarding the viability and up- and downsides with regards to sparse data and K-nns.

For the NGO we would recommend to look into the possibility of using the model as a tool when analyzing where previous and current priorities and problems lie. The way we used it for a few small experiments, is to see the impact of the different categories from one feature on predictions. This can be used to see whether overall priorities align with spending, i.e. if helping children is considered a priority it would show by having a higher spending and/or reduced variability of the spending. It can also be used to check if budgeting is less reliable in certain countries, to find what teams may need more help in this area, or to confirm ideas.

### 6.3.1 Limitations

One of the limitations during the research was the amount of processing power available to train the deep NNs. From experiments with the small data set strong evidence is found to support the expectations that performance of the model could be improved if it had more nodes per layer, and to a lesser extent if it had more layers. However in order to properly find good parameters too many configurations will need to be tested, which take too long to train for it to be feasible with the available resources. The expected gain is not such that predictions will significantly outperform those of the NGO, but the difference is estimated to be between one- and several hundred dollars on average per project, so this is not a neglectable amount either. If the model is to be implemented full scale we would recommend to try to find the resources to optimize the parameters for a larger model.

We did not do a full scale Principal Component Analysis to reduce the number of features however we did some less extensive exploration into leaving out individual features and groups of features. A PCA is designed to reduce overfitting, reduce redundant correlation and improve learning speed because of a decreased number of input features. A proper PCA may improve accuracy, although the increase it is expected to be limited. This can change if new features are added, and as such we would recommend to look at feature selection when revisiting these models after most of the changes to the NGO have been made.

# References


Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, 175-185.

Bengio, Y., & Faggella, D. (2019, February 19). *The Rise of Neural Networks and Deep Learning in Our Everyday Lives – A Conversation with Yoshua Bengio*. Retrieved August 22, 2019, from Emerj: https://emerj.com/ai-podcast-interviews/the-rise-of-neural-networks-and-deep-learning-in-our-everyday-lives-a-conversation-with-yoshua-bengio/

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer Science+Business Media, LLC.

Domingos, P. (2012, October). A few useful things to know about machine learning. *Communications of the ACM, 55*, 78-87. Retrieved from University of Washington: https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

Drucker, H., Burges, C., Kaufman, L., Smola, A., & Vapnik, V. (1996). Support Vector Regression Machines. *Advances in Neural Information Processing Systems 9*, (pp. 155-161).

Faggella, D. (2019, February 19). *What is Machine Learning?* Retrieved August 22, 2019, from Emerj: https://emerj.com/ai-glossary-terms/what-is-machine-learning/

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistic, Volume 29*, 1189-1232. Retrieved from https://projecteuclid.org/euclid.aos/1013203451

Frumkin, D. (2019, March 6). *RMSprop*. Retrieved August 28, 2019, from Golden: https://golden.com/wiki/RMSprop

Gonsalves, B. a. (2016). Improved web service recommendation via exploiting location and QoS information. *2016 International Conference on Information Communication and Embedded Systems (ICICES)* (pp. 1-5). Chennai: IEEE.

Hale, J. (2018, September 22). *Smarter Ways to Encode Categorical Data for Machine Learning*. Retrieved from Towards Data Science: https://towardsdatascience.com/smarter-ways-to-encode-categorical-data-for-machine-learning-part-1-of-3-6dca2f71b159

Hinton, E. G., Krizhevsky, A., Sutslever, I., & Srivastva, N. (2016). *United States Patent No. US9406017B2.*

Lawrence, S., Giles, C., Tsoi, A., & Back, A. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 98-113.

Leung, F., Lam, H., Ling, S., & Tam, P. (2003, January). Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, 79-88.

Lin, S., Zhang, Q., Chen, F., Luo, L., Chen, L., & Zhang, W. (2019, June). Smooth Bayesian network model for the prediction of future high-cost patients with COPD. *International Journal of Medical Informatics, 126*, 147-155.

*Machine Learning*. (n.d.). Retrieved August 22, 2019, from Cousera: https://www.coursera.org/learn/machine-learning

*Machine Learning*. (n.d.). Retrieved August 22, 2019, from Wikipedia: https://en.wikipedia.org/wiki/Machine_learning#Types_of_learning_algorithms

Michael Copeland. (2016, July 29). *What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?* Retrieved August 22, 2019, from Nvidia: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Mitchell, T. M. (July 2006). *The Discipline of Machine Learning.* School of Computer Science Carnegie Mellon University, Machine Learning Department, Pittsburgh.

*MNIST database*. (n.d.). Retrieved August 22, 2019, from Wikipedia: https://en.wikipedia.org/wiki/MNIST_database

Nikolic, D., & Faggella, D. (2018, November 29). *Welcome to AI Kindergarten, a Path to Machine Genomes*. Retrieved August 22, 2019, from Emerj: https://emerj.com/ai-podcast-interviews/welcome-to-ai-kindergarten-a-path-to-machine-genomes/

Pupil, P., Novovičová, J., & Kittler, J. (1994, November). Floating search methods in feature selection. *Pattern Recognition Letters*, 1119-1125.

Pyle, D., & San José, C. (2015, June). *An executive's guide to machine learning* . Retrieved August 22, 2019, from McKinsey & Company: https://www.mckinsey.com/industries/high-tech/our-insights/an-executives-guide-to-machine-learning

Salian, I. (2019, August 2). *SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?* Retrieved August 22, 2019, from Nvidia: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/

Sharma, S. (2017, September 6). *Activation Functions in Neural Networks*. Retrieved August 27, 2019, from Towards Data Science: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Silver, D. (Director). (2015). *RL Course by David Silver* [Motion Picture]. Retrieved November 2018, from https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PLzuuYNsE1EZAXYR4FJ75jcJseBmo4KQ9-&index=2&t=0s

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks fromOverfitting. *Journal of Machine Learning Research*(15), 1929-1958.

Theodoridis, S. (2015). *Machine Learning.* Elsevier.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society, Series B (Methodological) 58*, 267-288.

Vergouwe, Y., Moons, K., & Steyerberg, E. (2010, October 15). External Validity of Risk Models: Use of Benchmark Values to Disentangle a Case-Mix Effect From Incorrect Coefficients. *American Journal of Epidemiology, 172*(8), 971–980.

White, H. (1980, May). A Heteroskedasticity-Consistent Covariance Matrix Estimator And A Direct Test For Heteroskedasticity. *Econometrica, 48*, 817-838.

Yampolskiy, R. V., & Faggella, D. (2019, February 4). *Artificial Intelligence's Double-Edged Role in Cyber Security – with Dr. Roman V. Yampolskiy*. Retrieved August 22, 2019, from Emerj: https://emerj.com/ai-podcast-interviews/artificial-intelligences-double-edged-role-in-cyber-security-with-dr-roman-v-yampolskiy/

Zhu, M., & Gupta, S. (2017). *To prune, or not to prune: exploring the efficacy of pruning for model compression.* Cornell: ArXiv.