IMPROVING A PRECISION AGRICULTURE ON-FARM EXPERIMENTATION WORKFLOW THROUGH MACHINE LEARNING

by

Amy Peerlinck

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 2019

DEDICATION

I would like to dedicate this thesis to my family. None of this would be possible without the unrelenting support of my parents, Luc Peerlinck and Iris Scheirs, who were okay with their daughter moving across an ocean to pursue her studies. I will be forever grateful for the opportunities and love they have given me throughout the years.

ACKNOWLEDGEMENTS

My thanks go out to the Numerical Intelligence Systems Laboratory (NISL) at Montana State university. More specifically, I would like to thank Tyler Forrester, Jordan Schupbach, Jacob Senecal, Neil Walton, and Na'Shea Wiesner for their support. They were invaluable in helping with experiments and discussing ideas, culminating in better understanding of the problem itself and the implemented algorithms.

I would also like to thank my advisor, John Sheppard, for his guidance and constant faith. It is important to find an advisor that complements a students' work ethic. Dr. Sheppard made me work harder than I ever have, and it paid off. I am grateful for the opportunities he has presented to me, and I could not have asked for a better mentor.

Lastly, a thank you to the members of my committee: Bruce Maxwell and David Millman. As a computer science student with a background in linguistics, agriculture is mostly unknown to me. Dr. Maxwell was always very helpful when it came to explaining the mysteries of Precision Agriculture and providing me with resources to gain more insight. Furthermore, I thank the OFPE team for providing site-specific data.

I have had the pleasure of working with Dr. Millman as well as taking a course from him. Through these experiences I have learned a lot, and I would like to thank Dr. Millman for his helping hand whenever I needed it.

ACKNOWLEDGEMENTS – CONTINUED

This project was funded, in part, by MREDI grant 51040-MUSR12015-02 and by NSF grant 1658971.

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

LIST OF TABLES – CONTINUED

LIST OF FIGURES

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF FIGURES – CONTINUED

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF ALGORITHMS

ABSTRACT

Reducing environmental impact while simultaneously improving net return of crops is one of the key goals of Precision Agriculture (PA). To this end, an on-farm experimentation workflow was created that focuses on reducing the applied nitrogen (N) rate through variable rate application (VRA). The first step in the process, after gathering initial data from the farmers, creates experimental randomly stratified N prescription maps. One of the main concerns that arises for farmers within these maps is the large jumps in N rate between consecutive cells. To this end we successfully develop and apply a Genetic Algorithm to minimize rate jumps while maintaining stratification across yield and protein bins.

The ultimate goal of the on-farm experiments is to determine the final N rate to be applied. This is accomplished by optimizing a net return function based on yield and protein prediction. Currently, these predictions are often done with simple linear and non-linear regression models. Our work introduces six different machine learning (ML) models for improving this task: a single layer feed-forward neural network (FFNN), a stacked auto-encoder (SAE), three different AdaBoost ensembles, and a bagging ensemble. The AdaBoost and bagging methods each use a single layer FFNN as its weak model. Furthermore, a simple spatial analysis is performed to create spatial data sets, to better represent the inherent spatial nature of the field data. These methods are applied to four actual fields' yield and protein data.

The spatial data is shown to improve accuracy for most yield models. It does not perform as well on the protein data, possibly due to the small size of these data sets, resulting in a sparse data set and potential overfitting of the models. When comparing the predictive models, the deep network performed better than the shallow network, and the ensemble methods outperformed both the SAE and a single FFNN. Out of the four different ensemble methods, bagging had the most consistent performance across the yield and protein data sets. Overall, spatial bagging using FFNNs as the weak learner has the best performance for both yield and protein prediction.

CHAPTER ONE

INTRODUCTION

Precision Agriculture (PA) is a discipline that deals with optimizing the overall workflow of farms, be it the actual machinery, management, fertilizer, export, or increasing crop quality. The overall goal of PA is to increase net return while minimizing, or at least not negatively affecting, environmental impact. With the increase in data availability through means such as satellites, wireless sensors, soil measurement tools, and many more, PA has been growing rapidly in recent years. All this data lends itself to researching different strategies to improve crop management. Some of the emerging research in PA includes: optimizing irrigation [59], applying variable rate fertilizer [103], detecting weeds [122], and controlling plant disease [84].

However, farmers generally do not want to deal directly with all this data; they prefer a simple and low-cost solution [98]. A recurring problem is how to deal with all the available data efficiently and how to use it while developing useful but easy-to-use tool for farmers. To this end, commercial decision support tools focused on PA have been emerging more in recent years [1, 2, 5]. Such tools can take farmer and field specific data as input and provide the farmer with different potential solutions to optimize their net return. However, many of these tools are expensive (e.g. \$1000/year for an affordable solution [5]) and require special equipment, or a specialist has to come in to perform an analysis. Not all farmers can afford such approaches, and these solutions may not always translate well to farms in developing countries. Therefore, it is important to come up with a low-cost solution that is more easily adaptable to farms across the world. This is where the On-Farm Precision

Experimentation (OFPE) project comes into play.

1.1 On-Farm Precision Experimentation

The OFPE project limits itself to cheaply available data and focuses on a site-specific approach. This means that the process adapts to the farm as a whole while giving field-specific recommendations. The goal of the experiments is to determine how certain parts of a field react to different nitrogen rates, while taking previous years' yield and protein into account. A nitrogen prescription map is created based the discretization of the yield and protein data into bins and randomly prescribing the nitrogen with equal stratification over these bins. Stratification is a statistical technique that divides the population into sub-populations before sampling from each of these subpopulations. In our case, we divide the population based on the previous year's yield and protein values. We then assign an equal number of each of the desired nitrogen values across each of these subpopulations. The farmers then provide the resulting yield and protein after applying the experimental nitrogen rates. This process is repeated for multiple years. As more information is gathered, effectively creating a spatio-temporal model, a more accurate model for predicting and optimizing net return can be created. The function to calculate net return includes expected yield and protein, crop price, nitrogen cost, and overall fixed costs. Using this function, we can determine the nitrogen rate to be applied to certain parts of the field to optimize net return.

The project can be divided into two main parts: experimentation and net return optimization. The research performed for this thesis looked at a specific problem in each of these components. First, a software tool was created to ease the process of creating the experimental prescription maps. However, the initially generated maps can have large jumps in nitrogen rates between consecutive cells on the field. Such

large jumps can put strain on fertilizer machinery. Therefore, the farmers prefer the prescription maps to be as smooth as possible as far as nitrogen rates go. To solve this issue, a genetic algorithm (GA) was developed that looks at minimizing these jumps while simultaneously maintaining the stratification across the yield and protein bins.

The net return optimization has one crucial component that can strongly effect the overall results: the expected yield and protein values. This brings us to the second goal of this thesis: providing more accurate yield and protein predictions. Machine learning (ML) is a broad field of computer science that uses a learner for optimization of a problem or solving a task. In ML the goal is generally to learn from data, meaning that a task can be improved through experience, and this improvement is tracked by some performance measure [80]. ML can be split into two broad categories: supervised and unsupervised learning [47]. Supervised learning uses a set of training data that includes the values to be predicted, known as labels or targets. Unsupervised learning does not have labeled data available and focuses on self-organization of unlabeled data to determine useful properties of the structure. In between these two fields exists semi-supervised learning, which uses a small labeled data set to initially train a model and then uses that model to predict values for unlabeled data, which then get added to the training set. Our work focuses on supervised learning, which can be subdivided into classification and regression. Classification predicts a discrete label for a data point, whereas regression tries to predict a continuous value. The latter is the category into which yield and protein prediction fall.

There are numerous machine learning techniques that can be applied to regression problems. We look at a small portion of these available methods, focusing on techniques that have been shown to work well for other disciplines in recent years. The first method we considered was deep learning. To this end, a shallow feed forward neural network (FFNN) was compared to a deep stacked autoencoder

(SAE). FFNNs are considered to be "unstable" learners. This means that when training a neural network with the same tuning parameters but on a slightly different data set, the final model is likely to be different each time. Therefore, these models have a large predictive loss. In other words, a small change to the training set can have a large influence on the predictive model [15]. The results for the FFNN and SAE were compared to linear regression and non-linear hyperbolic regression. Second, two different ensemble methods, bagging and boosting, were applied using shallow FFNNs as the weak model. Ensemble methods combine multiple weak learners to provide the final prediction. Ensembles can be especially useful for neural networks, as it can improve the stability of the model [14]. Bagging [14] stands for bootstrap aggregation and randomly selects a subset of data with replacement from the original data set. This is repeated a desired number of times, and each of the resulting subsets is used to train a weak model. These weak models are then averaged to make the final prediction. Boosting [116] uses a similar technique, but instead of performing random selection with replacement, the data points are given weights based on how difficult they are to predict. The first model uses equally weighted data points; based on which of these data points were not correctly predicted, the weights are adjusted for the next model to be trained. The models are combined based on their overall predictive quality, meaning that a good predictor has a larger influence on the final prediction. More specifically, we implemented three different AdaBoost (or Adaptive Boosting) methods in our research: AdaBoost.R2, AdaBoost.RΔ, and a novel implementation combining aspects of each of these algorithms called Approximate AdaBoost. All models were applied to four different fields, each from a different farm. Furthermore, to further improve results, simple spatial sampling was implemented. This resulted in 16 different data sets: four yield, four protein, and the spatial variations of each of those.

1.2 Research Questions

There are a few different questions we are trying to answer; we will start by considering the objective of using a GA for rate jump minimization. Then we will describe the different ways we try to improve yield and protein prediction, as well as the expectations for each of the approaches.

The goal of the prescription software was to create a simple tool with several options to manipulate the prescriptions that are important to the farmer. The main research focus was not on the software tool itself, but on finding a solution to minimize rate jumps efficiently while maintaining the randomly stratified nature of the experiments. We propose a GA to provide this solution, as we believe the stochastic nature of the algorithm will sufficiently explore the search space and provide a satisfactory solution for both experimental purposes and the farmers' needs. Furthermore, a GA has the ability to take constraints into account in the form a fitness function, and this fitness function can be written as a multi-objective optimization problem. As the problem at hand is trying to balance between two independent problem statements, maintaining stratification across the bins and minimizing rate jumps between cells, the GA seems like a natural fit. The main question of this research is: Can a GA provide a more suitable prescription map that has smaller rate jumps while still maintaining random stratification of the experiments? We expect that some weight adjustments of the fitness function will be necessary to speed up the jump reduction. This is important for actual usage as a farmer does not have the luxury to wait for a long time when trying to download a prescription map. We also believe this could increase convergence rate, however, it might come at the cost of stratification.

The problem of yield and protein prediction carries a lot of weight, as these

predictions influence the overall net return optimization of the farms. Keeping this in mind, it is important to determine the best way to tackle this problem. The question whether there exists a single model that consistently performs better than the rest across all data sets is too broad. There is a large difference in size between the yield and protein data sets, and there are large differences in topology of the fields, making it difficult to find a single model that always has the "best" performance. However, it is possible that a single model performs well across all data sets, providing consistency in results, which would make it a viable solution for a real world application.

When comparing ML techniques to linear regression and the fit of a hyperbolic function, we hypothesize the ML approaches will outperform on all data sets. This is due to the inherent non-linear nature of yield and protein functions. As neural networks are known for their non-linear approximation capabilities, they are a natural choice for the yield and protein prediction problem. Furthermore, neural network parameters can be tuned to suit different types and sizes of data. Due to the large difference in data between yield and protein as well as between different farms, the adaptability of the neural network could be an invaluable tool. The main question here is whether a deep neural network will perform better than a shallow neural network? Generally, deep neural networks tend to improve prediction when compared to shallow neural networks. We expect this will hold true for yield prediction, but due to the small size of the protein data sets (300 to 3000 data points), a deep network may not perform as well when predicting protein.

In the context of ensemble learning, research has shown that combining many unstable learners improves performance [14]. As an FFNN is an unstable leaner, we can be fairly certain all of the proposed ensembles will improve results overall, due to the increase in stability compared to a single FFNN. The more important question then becomes which of the ensembles will have the best performance? Often boosting

outperforms bagging due to the added weighting of the models. We expect the same to hold true for this problem. Putting more emphasis on harder to predict yield and protein points is presumed to improve overall prediction quality.

A third question regarding yield and protein predictions now arises. As both deep learning and ensemble methods are expected to perform better than a shallow FFNN, which of these two approaches will perform better overall? As ensemble methods tend to reduce variance, and they are more generalizable, we study if both boosting and bagging have more consistent performance across the different data sets. Even if the deep network performs better on some of the data sets, we do not expect the ensemble methods to perform significantly worse. As we also think boosting will outperform bagging, boosting is expected to be the most consistently well-performing model.

The last question we try to answer is how does using surrounding data, and the resulting spatial data set, influence the results? We hypothesize that adding in this extra information will improve results for yield prediction across all models. However, due to the small number of protein points, adding in those extra features is expected to decrease predictive power.

1.3 Organization

The rest of the thesis is organized as follows. Chapter 2 covers relevant background on PA and related work that uses the same machine learning techniques in PA. It starts with a general overview of the field, and goes into more detail on subjects related to the topic at hand. The section on information management and decision support tools gives examples of commercial tools and talks about more experimental research. Fertilizer optimization covers the environmental impact as well as possible solutions. Yield mapping and yield prediction are both covered in

the yield experimentation section. The related work section discusses applying GAs to different areas in PA. Then the research area of yield prediction is looked at more closely, covering linear and non-linear models, as well as artificial neural networks, deep learning, and ensemble methods.

In the chapter on the OFPE workflow (Chapter 3), an overview of the entire project is given. The data sets are explained in more detail, followed by a description of the prescription software. The original linear and non-linear regression techniques and the spatial sampling process are explained in the data analysis section. Lastly, the economic model to be optimized is presented.

This brings us to the central components of the research, the algorithms. Starting with the prescription map optimization problem related to minimizing jump rates, Chapter 4 gives background information on GAs before detailing the implemented algorithm. This chapter also explains the two different bin discretization methods in more detail. Then a comparison is made between different GA implementations, different parameter settings, and the two discretization methods.

For the second part of our research, there are two different analyses for yield and protein prediction: deep learning and ensemble methods. Chapter 5 explains how FFNNs and SAEs are constructed and gives an overview of the experimental set-up. The neural network results are compared to the linear and non-linear models and the results are shown and discussed. The ensemble methods using the FFNN are explained in Chapter 6. This chapter also talks about related work concerning bagging and boosting of neural networks specifically. The methodology and selected results are discussed as well. An overview of all results for the different data sets is shown in Appendix C.

Because of the large amount of results, not all data is shown in detail in the deep learning and ensemble chapters. Furthermore, the two different types of methods need

to be compared as well. This is what Chapter 7 contains: a comprehensive overview of the results for each of the analyzed fields. This means results across all models and fields are discussed in more detail.

Finally, the concluding chapter summarizes our contributions and whether the hypotheses were confirmed. Furthermore, possible future work for each of the different research aspects is discussed.

CHAPTER TWO

BACKGROUND

2.1 Precision Agriculture

As an area of Study, Precision Agriculture (PA) originated in the 1980s with the idea of improving fertilizer application by changing the treatment based on the area of the field [97]. This idea of field specific application became possible thanks to the Global Positioning System (GPS) technology, which can determine coordinates anywhere on earth, enabling machines to apply treatments with localized requirements for each field [120]. Since then, it has become a much broader research field where various technologies are applied to improve agricultural outcomes. Furthermore, some of these techniques are applied to horticulture [104], viticulture [81], and livestock management [8] with similar improvements.

Overall, PA focuses on applying new technology to agriculture to decrease cost and increase net return, with the added goal of maintaining or improving environmental impact. Traditionally, this includes research in the areas of Geographic Information Systems (GIS) and GPS, which enable spatial analysis. However, GIS can be considered to be overly complicated for non-specialists. To this end, creating easier to use models that interact with GIS in the background became an important part of PA as well [97]. In recent years, technologies that have gained a lot of traction in PA are computer science approaches, such as machine learning [19] and software engineering [88], and different engineering approaches, such as remote sensing [83]. These technologies are applied to predict yield [6], create optimal water irrigation systems [130], provide farmers with fertilization prescriptions [59], improve machinery [131], and anything else that could benefit the farmer.

Taking all the different aspects of farming into account is not an easy task, especially with the wealth of data that has become available in recent years. Farmers can get overwhelmed by the numerous technologies available, and research shows that they want a product that is equally "useful" and "easy to use" [98]. Furthermore, simply applying new precision technologies is not enough to improve the entire farming production process [17]. For optimal advancement, the creation of user friendly, site-specific decision tools is necessary. Such a tool would gather data through on-farm experimentation and then use the gathered site-specific data for further analysis [77].

PA decision tools consist of several sub-processes, such as experiment design, yield prediction and economic optimization models. The following sections give a brief overview of decision tools in PA and explain the fertilization prescription and yield and protein prediction sub-processes.

2.1.1 Information Management and Decision Support Tools

Information management systems and decision support tools are pieces of software that enable farmers to make more informed decisions about their crops based on data analysis. Some of the issues with agricultural tools in the early 2000s were their usability and the use of scalar data. The interfaces were too complicated and contained too many unnecessary options. It is important to keep the part of the application that requires direct interaction with the farmer as simple as possible, as it is infeasible to perform large-scale training of farmers. The second issue at this time was the use of scalar data, whereas the standard in PA is to use GIS data formats, for example Well-Known Text (WKT) and Shape files [88]. However, in recent years, tools have become more sophisticated, and several commercial tools are on the market to help farmers with field management. (See [72] for an overview of decision support

systems in Precision Agriculture.)

A general data management tool is provided by Trimble Agriculture [5]. The company provides two different software bundles, Farmer Fit and Farmer Pro, for farm data management and record keeping. Aside from data management, the software also includes the option to create "profit maps" and track cost per unit of production for separate fields. Adapt-N [1] is a nitrogen recommendation specific system that found its roots at Cornell University. It uses crop, soil and weather data to provide point, zone, or 60×60 foot grid-based nitrogen recommendations. EFC Systems is an Agribusiness company that has several different services available for different users in the agriculture industry. MerchantAg and VanguardAg target retailers and distributors respectively. For farmers, the company has two tools: AgSolver and FieldAlitycs [2]. AgSolver [3] focuses on increasing net return by performing financial analysis of the field, whereas FieldAlitycs [4] is their precision tool, aiming to improve operational efficiency by providing field data management.

While all these tools can be useful, they are generally expensive (e.g. Trimble's cheapest option is \$1000/year [5]) and do not provide much transparency in their process. For example, the soil measurements used in many of these types of solutions require specific tools, and numerous measurements are necessary to obtain an accurate overview of a field's properties, resulting in an expensive overall process [106]. Adoption studies also indicate that farmers who use these technologies are well-educated farmers who own larger fields and have some affinity with computers and technology. These specific farmers are known as the "early adopters" [98]. These early adopters are only a small part of the farming community, and they have also reported issues with the use of PA on their farms. They note the lack of proper information usage and the need of many different processes to create an accurate overview of a field. Furthermore, these processes are either unrefined or technology intensive,

making them difficult to use [105]. To bridge the gap between farmer and technology, more straightforward and less expensive tools with a lower adoption threshold need to be implemented.

One important aspect that all modern decision tools have in common, is the use of field-specific data. Such field-specific management leads to better use of resources, e.g., through precision nitrogen fertilization and precision irrigation, which improves environmental sustainability while increasing crop yield [17]. Several studies have shown the benefits of site-specific irrigation, pesticide application, yield monitoring, and fertilizer prescription, among others [134].

Comprehensive, inexpensive decision-support tools for site-specific crop management are still lacking, especially systems that are adaptable across farms and allow the farmer a certain level of freedom [77]. Pedersen and Lind [94] discuss the progress in PA and conclude there is need for better information management and decision support systems, together with better integration of fertilizer plans into these systems. Thus, the importance and value of a well built and low cost site-specific decision support tool has become more apparent in recent years.

2.1.2 Fertilizer Optimization

There has been a substantial amount of work done in the area of yield monitoring and fertilizer optimization, as discussed in the surveys by [57, 62, 75, 100, 103]. Fertilizer application is an essential part of farming with a large influence on yield and net return. Due to the increase in agricultural production over the past 50 years, the amount of fertilizer being applied has increased significantly. This rise in fertilization results in a boost in production, but it also increases agricultural emissions, more specifically nitrogen emission, in groundwater and surface water, thus impacting the environment [22]. Excess nitrogen increases algae growth in water, harming water

quality and significantly decreasing oxygen availability, thus resulting in increased illness and death of aquatic lifeforms. An overabundance of nitrogen also has negative effects on soil nutrient balances and can encourage non-native plant species to grow instead of the desired crops [26, 89].

Agriculture is one of the main sources of nitrogen pollution. Not only from fertilizer application, livestock manure and soil erosion also play important roles [31]. These issues, combined with the goal to apply nitrogen more effectively to increase yield and thus profit, motivated research into the field of fertilization optimization [125]. Especially because decreasing the amount of nitrogen is a relatively simple way to lower costs that at the very least maintains crop profitability [12].

Bongiovanni and Lowenberg-DeBoer [12] give an overview of different fertilization studies performed in PA. The first area of research their literature review covers concerns nutrition management with a focus on research that studies how fertilization application improves yield. The authors found that most research concurs with the observation that variable rate application (VRA) for site specific management performs better than using a uniform application rate. Variable rate technology is also used in the experimental set-up of most studies. For example, a study on spring wheat in Argentina focuses on "split application" of nitrogen, meaning they apply N rates twice at different stages in the growing season [126]. However, the authors used a randomized block application of different nitrogen rates as their experimental set-up. The study found that if there is an excess of water at the beginning of the season, in combination with an adequate amount of moisture, split fertilization improves yield by 300 kg/acre. In all other cases, however, there seemed to be no significant differences. Furthermore, the authors noted that increasing the N rate above 90 had no significant impact on wheat yield.

VRA calculates optimized nitrogen rates for separate plots in a field instead

of applying a uniform rate across the whole field, and the optimized nitrogen is used as the basis for predicting return (i.e., yield, protein, or net profit). However, sprayers and spreaders that are able to apply these different rates are more expensive, and switching rates takes a higher toll on their mechanics, especially if the rate is changed frequently or if the variations involve particularly large changes. Therefore, it is no surprise that Bongiovanni and Lowenberg-DeBoer's survey [12] reports two studies where uniform rate application had higher profit [67, 128], especially since both of these studies were conducted when the equipment was less well developed. With the continuous improvement of farming equipment since the beginning of PA, VRA experiments have become more common practice. As a result, VRA has been used successfully in several more recent studies to improve efficiency of fertilizer application, thus reducing costs, confirming the initial theory proposed at the beginning of PA in the 80s [64, 103].

To show the widespread use of VRA, across crops as well as geographically, a short overview of different studies is presented. Kaizzi *et al.* look at nitrogen (N), phosphorus (P) and potassium (K) treatments for bean, soybean, and groundnut yield maximization in Africa. The authors increased mean yield by 92%, 111%, and 92% for each crop respectively using VRA of N to bean and of P to soybean and groundnut [59]. A study on potato growth in the Pacific Northwest determines the effects of uniform and variable rate technologies on nitrate contamination of groundwater [130]. They applied the two different techniques to adjacent fields in southeastern Washington and found that VRA reduces nitrate leaching. Variable rate technology (VRT) is also used for precision irrigation of crops. Nahry *et al.* [85] look at fields in Egypt to study the difference between traditional farming and precision farming techniques on maize. They apply VRT to both fertilization and water requirements of the crops. Their results show an increase in yield from 200 to

2100 kg/acre when using precision techniques. Precision irrigation is an important area of research in South African agriculture, as there is limited water available and a large area to cover. Dennis and Nell [28] propose a precision irrigation system in combination with VRA to increase crop yield. Their research indicates that most farmers in South Africa have not started using precision techniques, and part of their goal is to show the financial viability of investing in precision farming equipment. Their results confirm financial viability, showing that a profit would still be made even after deducting costs, as well as much more efficient water usage.

Even with these demonstrated gains, a lot of farmers still use uniform rate application. This can be explained partially by the lack of precise knowledge on what influences the choice for a specific fertilizer rate and the existence of several site specific factors often not being taken into account when creating variable rate prescription maps. To encourage farmers to implement VRA, more feasibility studies and a more in-depth analysis of the process would be beneficial. It is important to convince the agricultural community to adapt to this strategy, as it will decrease nitrogen application, benefit environmental safety, increase food production, and reduce overall production cost [25, 44].

2.1.3 Yield Experimentation

Yield is one of the most important factors for crops in farming. It ties directly into fertilization and net return optimization, and it is usually the main factor in both these processes. For Montana farmers, another important aspect of wheat production is the protein content of the grains. As Montana wheat is often used for bread production, farmers are trying to maximize the protein content of their grains as well as yield. Similar to yield, protein content is partially determined by genetic factors, but it can also be influenced by environmental factors [48].

There are two main areas in yield research: yield mapping and yield prediction. To be able to use yield data, it needs to be understood properly. This is where yield mapping comes in, which provides a visual representation of yield data. Yield prediction, on the other hand, uses different site-specific properties to determine the next season's yield across a field. Such predictions can then be used as another factor in fertilizer and net return optimization processes.

Yield mapping concerns the creation of a spatially referenced, graphical representation of crop yield for a certain area. These maps can be based on physical measurements or estimates of the yield using computational models to predict the values. Yield maps are generally split into four categories: inference, prediction, interpolation, and aggregation [96]. Inference maps associate yield estimates with existing map coordinates without changing the delineations on a base map, for example when specifying a yield goal with a soil map. Prediction maps fill in values on the map where a yield component is predicted rather than measured. Interpolation maps are coarser than prediction maps and are created by making yield measurements at specific locations within a defined area. Then the yield values between data points are estimated using some form of local estimation. Lastly, aggregation maps render aggregated statistics from the original data, either through measurement or prediction. Of these four types of yield mapping, the aggregation and the prediction of data are the most used in PA.

Essentially, yield mapping captures the mass or volume of grain or harvested crop by location within a specified field and requires three measurements: the yield measurement itself, the area over which the measurement was taken, and the location of the measurement within a field in that area [1] [96]. Yield mapping is closely related

[1] In our context, we also track and predict protein content in the harvested crop. While protein is not the same as yield, the associated map is essentially just another kind of yield map.

to our objective in fertilizer optimization, as it is often used as a basis to calculate the prime fertilization rate for fields.

Yield prediction tries to determine the function of yield response to environmental factors, such as field variables and weather, as well as crop data. A first challenge is identifying which factors strongly influence this yield function. And a second goal is to find a model that is capable of determining the best fitting function across many different fields under a plethora of circumstances. Often accurate yield estimation is possible when using soil samples, remote sensing, or molecular markers [13, 51, 65]. However, often this type of data is expensive to gather and therefore not a feasible solution for many farmers across the world. This is why research focusing on using freely available data, or data that has a low gathering cost, is important to the future of worldwide PA.

2.2 Algorithmic Methods in Precision Agriculture

This section explores related work in the different areas of research proposed in this thesis. Genetic algorithms are discussed in the broad field of precision agriculture, discussing several applications to different aspects of the field. Then related literature in the area of yield prediction is discussed for all the different models used in our research: linear and non-linear models, shallow and deep neural networks, and ensemble methods.

2.2.1 Genetic Algorithms in Precision Agriculture

Genetic algorithms (GAs) have seen limited use in the field of precision agriculture and do not appear to have been applied to the specific problem of minimizing rate jumps for experimental prescription maps. However, there have been successful applications of GAs to other aspects of PA. The following sections give a

brief overview of some of the studies using GAs or other population-based methods in several different areas of PA.

2.2.1.1 Fertilizer and Irrigation Strategies One study looks at using a multi-objective fireworks optimization technique for variable-rate fertilization. Their goal was to find the optimal fertilization prescription for oil crops based on yield, energy consumption, and environmental effects [136]. As their goal was to find the final optimized prescription map, their research did not look at minimizing rate jumps to decrease stress on the machines. Similar to this research, Lehmann and Finger [70] used an economic optimization model to determine the fitness of nitrogen prescription maps and irrigation strategies for optimal crop yield in western Switzerland. They also integrated climate change into their GA model and found that nitrogen rates can be adapted to minimize financial loss for farmers.

2.2.1.2 Sub-process Optimization Several PA studies have implemented GAs to optimize smaller parts of a larger process. What follows are some examples that use a GA to tune part of an algorithm. With the goal of correctly classifying crops and weeds, Noguchi *et al.* [90] implemented a fuzzy logic system. A GA was used to fine-tune the fuzzy logic membership rules. Once the weeds have been separated from the crops using this technique, an artifical neural network was used to estimate crop height and width. Timlin *et al.* [123] estimated water holding capacity (WHC) of a field based on yield map data. The field was transformed into a grid and a GA was used to optimize the WHC for each grid cell. The resulting grid was then used to classify a field into areas that are more or less susceptible to droughts.

GAs have also been applied to transform the input data for yield prediction. In order to explain variability and uncertainty in crop yield based on soil sampling density and weather data better, Pachepsky and Acock [92] used a GA to perform

stochastic imaging of available soil water capacity. The results were then used to estimate variability and uncertainty in crop yield for a soybean crop model. Natarajan *et al.* [86] used a fuzzy cognitive map (FCM) model, represented by a directed weighted graph, for yield classification. The initial map was based on concepts determined by experts. The resulting weight matrix was then updated using a GA to find the final FCM weight matrix. This finalized matrix was the input for the final classification algorithm.

2.2.1.3 Wireless Sensor Networks An important technology in the field of PA is the use of wireless sensor networks (WSN) to monitor plant health, climate, and other field specific features. Such networks provide useful information to the farmer, but they require a lot of energy. Below, we discuss several studies that have used a GA to find the optimal sequence of sensors to improve energy consumption.

In order to optimize battery consumption of the sensors, Ferentinos and Tsiligiridis [38] created a fitness function taking application-specific, connectivity, and energy-related parameters into account. Through the use of the GA, the authors discovered that it is more desirable to have a larger number of active sensors with lower energy consumption, as opposed to a smaller number of sensors with higher energy consumption.

Liu *et al.* [74] used four different WSN deployment criteria for their fitness function: node location in corresponding plots, k-connectivity of network, no communication silos in the WSN, and definition of a minimum distance between nodes and the plot boundary. The authors found that their approach resulted in a network using less nodes than regular patterns such as hexagon, square, or triangle patterns.

A comparison of five different GA implementations for WSN optimization was

made by Mirhosseini *et al.* [79]. To determine the best model, they measured the number of measuring cycles a WSN stays alive, where the measuring cycle is a specified timespan. If a network stays alive for more measuring cycles, it means that it has longer battery life. The five implemented algorithms are: Binary Genetic Algorithm, Binary Particle Swarm Optimization, the Quadrivalent Quantum-Inspired Gravitational Search Algorithm, the Binary Quantum-Inspired Gravitational Search Algorithm (BQIGSA), and an elitist version of the latter. The network created by the elitist BQIGSA remained active for the most cycles.

2.2.1.4 Crop Management A subfield of PA is weed detection and crop management, which tries to eliminate weeds from crops. Neto *et al.* [87] considered methods using GAs for plant classification based on foliage. Successfully identifying whether a plant on a field is a weed through aerial images can make it easier to remove unwanted weeds from a field. Their method performed leaf extraction by clustering leaf fragments, obtained from pictures, and creating a chromosome out of a leaf fragment and its neighbors. A GA was applied to reconstruct the fragments of non-occluded, individual leaves. These reconstructed leaves were then used for plant identification and mapping to improve weed control and crop management. In the same vein, Tang *et al.* [122] performed weed sensing based on color image segmentation, where the GA was used to identify specific regions in a Hue-Saturation-Intensity color space to segment plants from background.

Crop management also deals with identifying sick plants; in a recent study hyperspectral images were used to identify charcoal rot in soybean stems [84]. A GA was used to select specific wavelengths from the hyperspectral image to improve classification. Similar work is also being done in selecting wavelengths and generating associated filter parameters for multi-spectral imaging in produce classification [127].

2.2.2 Yield Prediction

As discussed previously, yield prediction is an important aspect of PA and has been widely researched since the beginning of the field. Initially, simple linear and non-linear regression models were applied to the prediction problem. However, often such models are limited in performance due to the complex nature of yield functions. In more recent years, Machine Learning (ML) has enabled more in-depth research in a plethora of fields. Training artificial neural networks (ANNs) is one of the most popular techniques in ML and has been applied to a variety of biological and agricultural problems [113]. An interesting and useful aspect of ANNs is their ability to find complicated associations between input and response variables without pre-defining any constraints or assumptions about the sample distribution of the data. This enables the possibility of describing complex non-linear relationships that are present in domains such as PA resulting from a wide range of crop conditions and other influencing factors [124]. Furthermore, Deep Learning has become a topic of much interest as it has been shown to improve the performance over shallow neural networks even further [114]. Similar improvements have been achieved by using ensemble methods, such as bagging and boosting [29].

2.2.2.1 Linear and Non-Linear Regression

Stepwise multiple linear regression (SMLR) is one of the most commonly used methods to develop empirical models from large data sets. Osborne *et al.* [91] used SMLR to estimate grain yield in nitrogen- and water-stressed corn. More specifically they used the REG procedure in SAS, a general-purpose regression procedure [7], to predict plant nitrogen, biomass, grain yield, grain nitrogen concentration, and chlorophyll meter readings.

Uno *et al.* [124] observed that there are three noteworthy drawbacks to using SMLR. First, SMLR is based on the assumption that a linear relationship exists

between input and response variables. Second, noise in the samples is assumed to follow a normal distribution, and third, in some cases, the model tends to overfit, thus reducing its applicability to unseen data. Furthermore, the research explores whether ANNs perform better than SMLR. Surprisingly, their results show that ANNs and SMLR perform similarly. Because of their inconclusive results, the authors emphasize that more research needs to be conducted using both models. In recent years, there has been numerous research comparing linear regression models to other statistical and machine learning methods [6,62,71]. A broader overview of some of these studies is given in the following sections.

Several older studies look at the relation between yield and weeds, and find that a hyperbolic regression model can model this relationship adequately. Cousens [20] compared different models to describe yield loss as a function of weed density. He found that a hyperbolic model fits the data better than other non-linear models, such as a sigmoidal function. Spitters *et al.* [119] modeled the relation between yield and weed density using this hyperbolic relation. They then linearized this model to perform linear regression to predict yield; however, they found that applying a logarithmic transformation reduced bias and improved predictive results. Rasmussen [102] used the same hyperbolic yield-density relationship to determine yield response to harrowing of weed. The results indicate that weed-killing may create crop damage that negates the effect of the harrowing.

2.2.2.2 Artificial Neural Networks This section focuses on agricultural uses of ANNs, more specifically on research concerning yield prediction and fertilization optimization. For a more general overview of ANN applications in biology and agriculture, Samborska *et al.* [113] summarize research where ANNs have been applied successfully.

Drummond *et al.* [33] used different methods of training ANNs on the same dataset to predict yield. They focused on training accuracy, test set generalization, and outlier rejection properties to determine the preferred method. The authors looked at backpropagation, with and without weight decay, as well as quickprop and rprop for their training techniques. Contrary to what Uno *et al.* [124] found, Drummond *et al.* showed that any form of learning and training outperformed the linear method, and that rprop as a learning technique performed slightly better than backpropagation. There were, however, several issues with the estimated yield map for all the used techniques. The authors hope that more research in this area will help eliminate these issues.

Since the aforementioned research, applying ANNs to PA has seen vast improvements. As mentioned previously, Uno *et al.* [124] performed a comparative research in 2005 where they state that there is more promise in ANNs than in linear models, especially for creating yield maps, even though their results for both models were very similar. This belief comes from the ability of neural networks to adjust the number of processing elements and network connections, making it more adaptable to different problems.

Dahikar and Rode [24] applied a feed-forward neural network trained via backpropagation to predict which crop would be best for a certain field based on different soil and atmosphere parameters. Their input parameters included pH-levels, nitrogen, potassium levels, temperature, and rainfall. Furthermore, they predicted the best fertilizer rate for the crop. Their predictions were intended for farmers who are not able to perform expensive soil tests. The results showed relatively accurate predictions and could hence assist these farmers.

In wheat yield prediction research specifically, ANNs have been applied successfully by Pokrajac and Obradovic [99]. They created software to improve net

return by maximizing an objective function including yield, crop price, and fertilizer cost. They performed simultaneous optimization as well as inverse modeling. Both these techniques require a regression estimation of the yield function. The authors implemented least squares linear regression as the basic model and compare it to a multi-layer feed-forward neural network (FFNN) with sigmoidal and radial-basis (RBF) activation functions. They applied their models to simulated wheat data and found that both types of neural networks increased net return compared to linear regression. More recent research by Ruß *et al.* [111] used neural networks to predict wheat yield from freely available and usually low-quality in-season data. Furthermore, the authors applied data mining with neural networks to optimize fertilizer usage. Different networks were built and evaluated, and the results showed that the prediction accuracy of the networks rose once more data became available. Alvarez [6] performed a study looking at the effects of climate and soil properties on wheat yield in the Argentina Pampas region. Surface regression and an ANN were implemented for yield prediction. Four factors were found to have a significant influence on yield: soil available water holding capacity, soil organic carbon, rainfall over crop potential evapotranspiration, and the photothermal quotient. Furthermore, the ANN obtained a statistically significantly lower RMSE than the regression model ($p = 0.05$).

Kaul *et al.* [61] used ANNs to determine corn and soybean yield for farms in Maryland under typical climate conditions, specifically precipitation. The results using the ANN were promising compared to multiple linear regression (MLR). A similar study was performed by Li *et al.* [71] for counties in the corn-belt in the Midwest of the U.S. They used the Normalized Difference Vegetation Index (NDVI) values to characterize the entire growth process of the corn and soybean crops. Two different NDVI data sets were used: MODIS and AVHRR. The authors came to

the same conclusion as Kaul *et al.*, namely that an ANN outperforms MLR, with the added conclusion that NDVI values were a useful source of information for yield prediction. ANNs were used by Panda *et al.* [93] to look at the influence of including four different spectral indices—NDVI, green vegetation index (GVI), soil adjusted vegetation index (SAVI), and perpendicular vegetation index (PVI)—for corn yield prediction. The authors found that using PVI gives more favorable results than the other indices and confirm the usefulness of using ANNs for yield prediction. Raw image data of apple fruit and tree canopies were used by Cheng *et al.* [18] for early yield prediction. The images were split into fruit, foliage, and background through image segmentation techniques and the resulting features—cross-sectional areas of fruit, small fruits, and foliage, as well as number of fruits—provided the input into the FFNN. The FFNN, in combination with the extracted features, was shown to achieve desirable results for early yield prediction of fruit [18].

2.2.2.3 Deep Learning A survey was conducted by Kamilaris *et al.* [60] on deep learning applications in PA. The authors identified 40 papers applying different deep neural networks (DNN) to a plethora of agricultural or related problems. The authors found that the most popular areas of research involve weed, plant, crop type, land cover, and fruit recoginition or identification. These problems all involve image processing; it is no surprise that the most common deep architecture used were different implementations of the convolutional neural network (CNN). A succesful application of a CNN to fruit detection was performed by Sa *et al.* [112]. The authors used transfer learning to adapt the Faster Region-based CNN to the fruit images using both RGB and near-infrared modalities. They trained the model to detect seven different fruits and achieve F1-scores that 0.8 for all seven fruits, and more than 0.9 for five of them.

There has been some limited research into using DNNs for yield prediction. You *et al.* [132] used remote sensing data, which is available worldwide, to predict soybean crop yield in the U.S. In their research, multi-spectral images were transformed into histograms of pixel counts of the wavelengths, effectively performing dimensionality reduction, and were then used as input to CNNs and Long-Short Term Memory networks (LSTMs) for yield prediction. In addition to these ANNs, a Gaussian process layer was applied to account for spatio-temporal dependencies. Their method outperformed decision trees and a 3-layer neural network for large-scale yield prediction by 15 percent on average. Kuwata and Shibasaki [66] compared a support vector machine (SVM), an auto-encoder (AE), and a deep neural network with six hidden layers for corn yield estimation across large areas. They found that the deep neural network had the most consistent performance (RMSE scores around 18 bushels per acre), and that DNNs and AEs did a better job than the SVM at extracting features from data. Recurrent Neural Networks (RNNs) with LSTM cells have been implemented by Jiang *et al.* [58] as well as Hu and Wang [57] to predict corn yield at a county level. Both papers used similar techniques by incorporating temporal weather data and remotely sensed data. The predictions made by the RNN were compared to data provided by the U.S. Department of Agriculture (USDA). They found that their predictions were generally closer to those made by the USDA.

CNNs have also been applied to satellite images, obtained from NASA's MODIS payload instrument, for crop yield prediction [109]. Two different CNN approaches were implemented in this research: a temporal and a spatio-temporal approach. The temporal approach created histograms from the remotely sensed weather and crop health data and sent those histograms through a CNN. The spatio-temporal approach added in extra spatial data layers, including soil properties and elevation, to the histogram CNN. The model consists of convolutional blocks, each followed by a max-

pooling layer and using the Rectified Linear Unit (ReLU) activation function. The authors compared their results for both CNNS to a decision tree, Ridge regression, and the Deep Neural Network proposed by Kuwata and Shibasaki [66]. The results showed that the spatio-temporal approach outperformed the histogram CNN as well as the three other models.

2.2.2.4 Ensemble Methods Yu *et al.* [133] introduced an ensemble of feedforward ANNs in fertilization models. They claimed there are two common issues when using single feedforward networks in this context. First, they found that assigning a maximum target yield before computation led to a large error in determining the fertilization rates. Second, they claimed that using a single ANN as their model led to limited forecasting accuracy and generalization capacity. The authors proposed using an ANN with nutrient concentration and fertilization rate as input and yield as output. They trained several neural networks via backpropagation using a bagging procedure and used k-means clustering to cluster the networks. Next they selected a representative network from each cluster to form an ensemble of ANNs. They determined the combining weights for the ensemble by applying the Lagrange multiplier method where the associated constraints enforced that the weights sum to one. Then they applied the ensemble to define a nonlinear objective function based on the ensemble output and optimized the fertilizer application rates through nonlinear programming.

Two different decision tree ensemble methods, boosted regression trees and random forests, were applied to winter wheat yield prediction in northern China [54]. Time-series NDVI values were used as input to the models. No significant difference in performance between either of the ensemble methods was found. An ensemble of ANNs was applied to predict sugarcane yield for Sao Paulo State in Brazil [39].

NDVI as well as MODIS time series data were used as input to the ensemble. As the first step, feature selection was performed through a wrapper process with sequential backwards elimination. Then the resulting features were split into N subsets, where each subset was trained separately using an ANN. The results were combined using a stacking ensemble method. The results showed that feature selection and smaller stack sizes outperformed the use of the full data set and combining a large number of networks.

Ruß and Henning [110] used five different regression techniques for yield prediction to determine which variables were most important for yield prediction. They used the regression implementations of support vector machines, decision trees, bootstrap aggregation (bagging) using decision trees, and random forests, as well as simple linear regression. The authors found that bagging, random forests, and linear regression had similar RMSE results. Khanal *et al.* [62] compared five different machine learning methods (Random Forest, ANN, Support Vector Machine, Gradient Boosting, and Cubist) to linear regression for soil property and corn yield prediction. The authors found that the different ML models performed better than linear regression overall. For yield prediction specifically, the random forest and cubic model had the best performance.

CHAPTER THREE

ON-FARM PRECISION EXPERIMENTATION WORKFLOW

Our overall research goals are focused on developing strategies for improved crop production. To this end, a project focusing on on-farm experimentation was initiated in Montana, with the goal of creating an economic-based decision model to increase yield, decrease cost, and improve environmental impact. The project involves analyzing farmer provided data for specific fields, including previous years' yield and protein, and freely available data on the field and crops (e.g., elevation, slope, topological position index, precipitation, and the normalized difference vegetation index (NDVI)).

All of this data is combined and analyzed to create site-specific experiments and to perform economic optimization. The experiments described in this thesis aim to create a nitrogen prescription map based on provided yield and protein information for a specific field and to generate models for predicting crop yield and protein production based on field data. For the first experiment, the yield and protein points are discretized into bins, and different nitrogen rates are evenly prescribed across the cells in each of these bins. In this research a GA is applied to find the optimal experimental prescription map; thus, this work falls generally in the area known as "design of experiments" [55]. In Figure 3.1 this is described as the "Experiments" stage.

The farmers then use these experimental prescription maps to apply nitrogen to their fields, and provide us with the resulting crop data, which is then added into the workflow. This experimental process is repeated each year to gather temporal data on the fields to improve predictive ability of yield, protein and net return (although,

without necessarily applying random application over the entire field), effectively creating a spatio-temporal data set.

So far, we have been describing the need for prescriptions to perform the necessary experiments to derive models for optimizing crop production. However, there is a second type of prescription map, resulting from performing the optimization based on these models. The resulting prescriptions then have the goal of yielding the optimal crop production, rather than generating data for modeling building. ("Field profit maximization" in Figure 3.1.) To determine the "optimal" prescription map, an economic model is used to optimize net return based on probability distributions of nitrogen cost and crop prices, as well as yield and protein predictions. As net return optimization is the ultimate goal, achieved by minimizing nitrogen costs and optimizing yield, accurate predictions of yield and protein are an important aspect of the problem.

This is the second goal of our research: improved yield and protein prediction as part of the "Data organization & analysis" step in Figure 3.1. The initial process used simple linear and non-linear models for yield and protein predictions. In our research, these predictions are improved through the use of spatial data in combination with several machine learning techniques.

3.1 Data Used

Four different yield and protein data sets were chosen, each representing a different field from a different farmer, totaling eight data sets. Prescription maps of each of the four fields can be seen in Figure 3.2. An overview of the number of points for each of these data sets is given in Table 3.1. For each yield and protein point, the data included the following features:

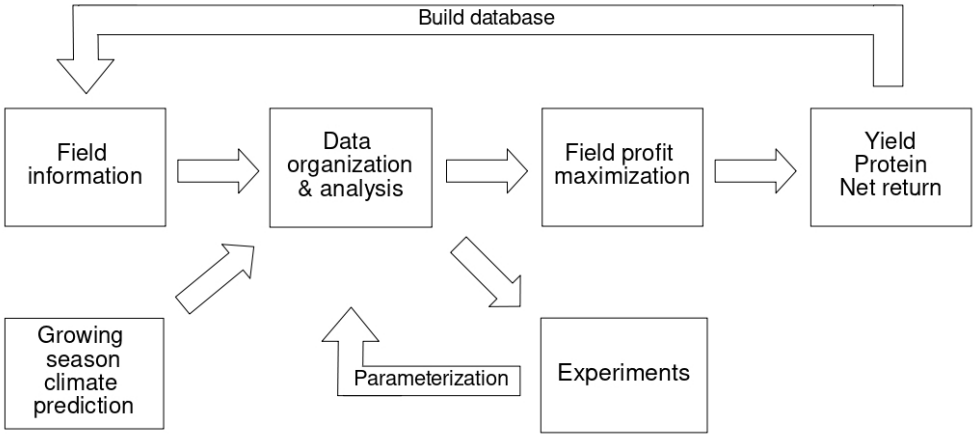


Figure 3.1: Flowchart of the optimization process for field profit maximization.

- Previous years' nitrogen application: After giving the farmers a prescription map, they use this to apply nitrogen to their fields; however, the nitrogen that is applied in practice often differs from the prescribed nitrogen. The latter is what we use in our data set.
- Field slope: The steepness of the terrain at a certain point, given in degrees.
- Field elevation: The elevation at a certain point, given in meters.
- Topographic Position Index (TPI): Quantification of the terrain water capturing potential surrounding the data point, based on elevation measures within $10m$ of each yield point.
- Field aspect: The degree a specific point is north and south facing is the cosine of aspect, whereas the east or west facing degree is the sin of aspect.
- Precipitation: The amount of inches of rainfall that year.
- Normalized Difference Vegetation Index (NDVI): A hyperspectral measuring unit to determine plant health.

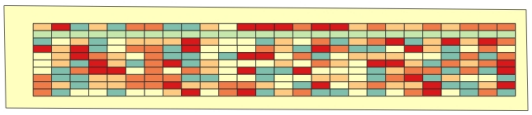
Table 3.1: Number of yield and protein data points for each field.

	sre1314		**sec35mid**		**davidson**		**carlin**	
	Non-Spatial	**Spatial**	**Non-Spatial**	**Spatial**	**Non-Spatial**	**Spatial**	**Non-Spatial**	**Spatial**
yield	24646	15868	17873	11589	11802	9062	15621	12775
protein	2998	1731	1019	558	560	311	655	457

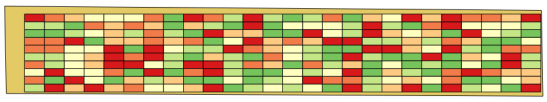
- Growing Degree Day (GDD): A measure of heat, used to determine how much time is needed for the crop to reach maturity.

The yield and protein values are what the implemented models are trying to predict based on the other information provided. The yield and protein values were obtained directly from the farmer and were from fields from the previous harvest. The number of protein points is at least an order of magnitude less for the protein data sets, as the protein monitors mounted on the harvesters were unable to sample at as high of a rate as the yield monitors. Yield is measured in bushels per acre, for winter wheat this translated to 60 lbs of wheat per bushel. Protein is represented as a percentage of the total grain.

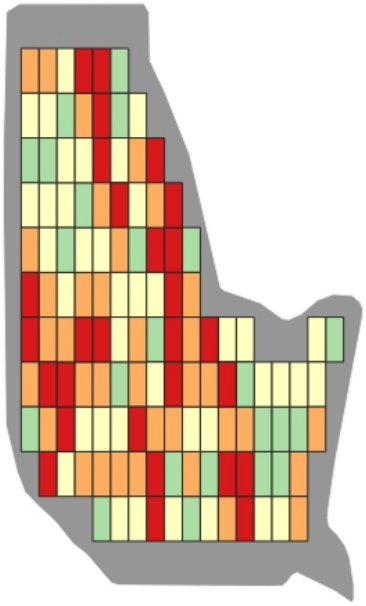
The amount of data for protein and yield points in each field varies greatly. Furthermore, the number of data points is reduced during the spatial sampling process (denoted as "Spatial"). To gather spatial information, the grid structure of the field is used, and only points in a grid cell with 8 neighboring cells (i.e. no field boundary cells) are taken into account, not all points are used to create the spatial data set. The difference this makes in data set size can be seen in Table 3.1.



(a) Field “CarlinWest”.



(b) Field “SRE”.



(c) Field “DavidsonMidWest”.

(d) Field “Sec35Mid”.

Figure 3.2: Unoptimized prescription maps of the four fields used in our research. Fertilizer levels in terms of pounds of nitrogen per acre go from low to high according to the red (low) to green (high) color spectrum.

3.2 Prescription Tool

The creation of nitrogen prescription maps can be a strenuous process. To facilitate this part of the workflow, a simple software application was created (see Appendix A for screenshots of the application). The first part of the application requires input from the user. There are several different options available to adjust the resulting prescription map, depending on the need of the farmer. Once the initial prescription is created, the map is shown on the user’s screen. The user is capable of

adjusting the initial input in the main window and create an entirely new prescription, or they can choose to adjust a single cell's value. Once a satisfactory map has been created, it can be downloaded as a CSV file. What follows is an overview of the different functionalities of the application.

In its simplest form, the application takes a previous year's yield file, the field shape, how many yield bins the data should be discretized into, the desired nitrogen rates (up to 6 values), the desired application plots' height and width (also referred to as "cells"), and a boundary buffer, indicating the minimum distance each of the cells should have from the field boundary. This will create a brand new map based purely on yield bin discretization, where the default stratification strategy is set to equal yield values, meaning the yield bin boundaries are decided by the actual yield values. Another option for bin discretization uses the number of data points, which means each bin has the same number of cells. The desired nitrogen rates will then be assigned evenly across the bins with no rate jump optimization.

In case the farmer also wants previous protein values to be taken into account, a file containing protein points can be uploaded, and the desired number of protein bins can be indicated. If there is both yield and protein data available, the total number of bins will be a combination of the desired number of yield and protein bins. For example, if the user wants the data to be discretized into three yield bins and two protein bins, the total number of bin combinations would be six. The nitrogen rates will then be spread evenly across these six combinations.

If the field for which the prescription is being generated already has a grid layover available, this grid can be uploaded. This feature provides consistency of prescription layouts over the years. Such consistency improves both the spatial and temporal aspects of the experimental analysis. Another useful feature for the experimental layout is the ability to create a strip trial instead of a grid trial. This means lengthwise

strips are being created across the field, as shown in Figure A.4 in Appendix A.

Lastly, the user has the option to minimize rate jumps between consecutive cells. To enable this function, an as-applied file has to be uploaded. Such a file contains information on how the spreader or sprayer applied nitrogen in previous years. The data points are sorted by time applied, and can then be used to determine what order the cells are visited by the spreader. As a result, the prescribed nitrogen rate for cells can be organized in such a way that the values between consecutive cells are not too far apart, thus reducing strain on the machines. This jump minimization is accomplished by the genetic algorithm described in Chapter 4.

3.3 Data Analysis

An important part of this thesis is the data analysis for yield and protein prediction. Our research focuses on applying different machine learning methods to improve prediction accuracy (Chapters 5 and 6). However, in order to evaluate the effectiveness of these methods, a baseline has to be included. Previous research in this project used linear and non-linear models to predict yield and protein, and these models will be discussed in more detail in this section.

Furthermore, a simple spatial analysis was performed to create new spatial data sets from the original data sets. These spatial samplings are then used in each of the proposed algorithms in the hopes of further improving predictions. The process used to create these spatial data sets is also explained in this section.

3.3.1 Linear and Non-Linear Regression

The basic data analysis, performed in the works by Maxwell *et al.* [76] and Lawrence *et al.* [68], uses linear and non-linear regression models. Multiple regression is a tool for statistical analysis that uses mathematical methods to define possible

relationships between variables. In linear regression, a straight line (or hyperplane) is fit through the data in order to best predict the response values. Often this method is used to predict certain values for a variable given another variable, but can also assist in explaining certain values or to test a scientific hypothesis [117]. The linear relationship used here models yield based on applied nitrogen rate, precipitation, NDVI, elevation, slope, and aspect sine and cosine:

$$\begin{aligned} Yield = \beta_0 + \beta_1 \cdot \text{Nrate} + \beta_2 \cdot \text{NDVI} + \beta_3 \cdot \text{precip} \\ + \beta_4 \cdot \text{elev} + \beta_5 \cdot \text{slope} + \beta_6 \cdot \text{asp}_{\text{sin}} + \beta_7 \cdot \text{asp}_{\text{cos}}, \end{aligned}$$

where the β_i's are parameters known as partial regression coefficients.

Nonlinear regression works similarly to linear regression but instead of constructing a linear surface, other surface shapes are implemented. This shape is determined by the overall spread of the data and can vary greatly. The implemented non-linear model in this research uses a hyperbolic function to estimate the yield and protein values. This choice is based on the idea that yield has a saturation point, a point at which it no longer increases [20]. The general hyperbolic fit used in this research can be modeled as follows:

$$Yield = \alpha + \frac{(\eta - \alpha)Nrate}{\frac{1}{\gamma} + Nrate},$$

where η represents a linear function defining the relationship between yield and previous year's yield, current NDVI, and two previous years' NDVI values. It defines the upper limit of yield. *Nrate* indicates the actual nitrogen rate applied at a certain point. α indicates the yield at 0 nitrogen rate, or the intercept, it represents the

geographic variables as a linear function of the sine and cosine of aspect of the point, slope, elevation, and TPI. Lastly, γ denotes the slope of the yield increase [76].

3.3.2 Spatial Sampling

In the yield and protein prediction process applied in this thesis, spatial sampling is applied to all implemented models. Spatial sampling is a process where observations are collected in a two-dimensional space. Usually, the sampling method is designed to capture as many spatial variations of the variables that are to be studied. After initial data has been sampled and documented, additional measurements may be made depending on the variations within the data. Generally, these additional measurements are made based on criteria to optimize the data [27].

The final collected data can be irregular due to a higher interest in certain variables of particular areas or due to terrain conditions. This irregularly spaced data is significantly more complicated to analyze than regularly spaced data. Therefore, managing these irregularities becomes important in order to analyze the data sets more efficiently [43]. One way to address this problem is to take surrounding data into account to create a spatial data set.

In our research, a field is laid out in a grid pattern; thus, a form of spatial sampling naturally arises from the grid-based arrangement imposed by the fields studied and the prescription maps defined for those fields. Data points reside in each grid cell; therefore, it would be natural to sample, not only from the target cell, but from the neighboring cells. Two common neighborhood functions for such sample are the von Neuman neighborhood (Figure 3.3a) and the Moore neighborhood (Figure 3.3b).

The von Neuman neighborhood only looks at the cardinal directions, whereas the Moore neighborhood includes the primary intercardinal directions. To use as

(a) Von Neuman neighborhood. (b) Moore neighborhood.

Figure 3.3: Neighborhood configurations for sampling. The center cell is denoted "C."

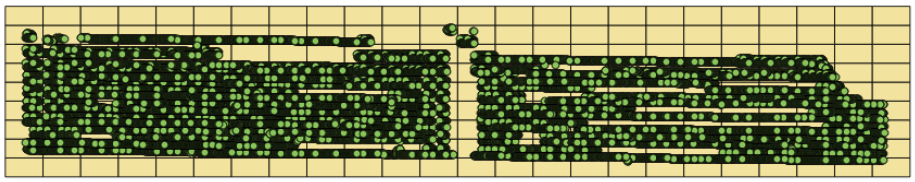


Figure 3.4: An example of a distribution of yield points, rotated at a 90 angle.

much information as possible, the Moore neighborhood was chosen as the spatial sampling method for the results reported here. Specifically, the average values of all points in a single cell were calculated. For each data point the averaged values of each of the 8 neighboring cells individually provides the spatial information for that data point. For more consistent spatial information, two alterations to the data were used to decrease noise and smooth the data. The first was the aforementioned averaging technique and second was to increase the grid cell size to contain more points in each cell. An example of a grid overlay on field sre1314 can be seen in Figure 3.4.

3.4 Economic Model

The economic analysis uses a probability distribution of crop prices, yield values and nitrogen cost to calculate the net return for a specific cell in a given year to

perform field profit maximization. Predictive models are updated annually using a Bayesian updating process. This spatio-temporal probabilistic Bayesian framework approach was shown to increase net return by \$23–\$25/ha [68].

Net return is measured in $\$/ha$ and depends on several different factors: Expected yield (E_{yield}); expected protein, which can have a positive or negative effect, resulting in a premium or dockage to the crop price (i.e. $Price_{crop} - Dockage_{protein}$); the applied nitrogen rate ($N_{applied}$), which is measured in kg/ha; and the fixed costs of the farm (FC). This results in the following net return function:

$$NetReturn = (Price_{crop} + Premium_{protein}) \cdot E_{yield} - Price_N \cdot N_{applied} - FC,$$

The overall goal is to optimize this net return function, and predicting yield and protein can be a useful improvement to achieve better optimization. For example, changing the nitrogen rate to be applied will result in different yield and protein predictions, thus changing the net return. Therefore, optimizing this function using the different results can help decide which nitrogen rate would be the best choice for a specific field.

CHAPTER FOUR

PRESCRIPTION MAP OPTIMIZATION: A GENETIC ALGORITHM APPROACH

As mentioned in previous chapters, one component of our research involves examining variable nitrogen rate application for winter wheat in Montana to optimize field profits through experimental treatments based on field specific data. A flowchart of the entire process is shown in Figure 4.1, the "Experiments" stage is emphasized, as the prescription maps are generated in support of the experiments. Yield and protein information from previous years are taken into account to generate a variable rate nitrogen prescription map. The available yield and protein values are discretized into bins, and each nitrogen rate is distributed evenly across each bin through random, stratified assignment. This results in a randomly stratified nitrogen prescription map (Figure 4.2), which can then be used to design an experiment allowing for proper analysis on how different nitrogen rates affect parts of the field with different protein and yield values.

When creating the experimental design prescription maps, as a result of the randomization process, there may be a large difference in the amount of nitrogen to be applied from one cell to another, putting strain on the farmer's equipment. To lessen the wear on the machines, these differences, or jumps, need to be minimized, resulting in a more gradual prescription map. This problem and an example of a more desirable prescription is illustrated in Figure 4.3.

In this chapter we investigate the application of a genetic algorithm (GA) to generate random, stratified nitrogen applications while simultaneously attempting to minimize the magnitudes of the jumps between the cells in the field. This means that

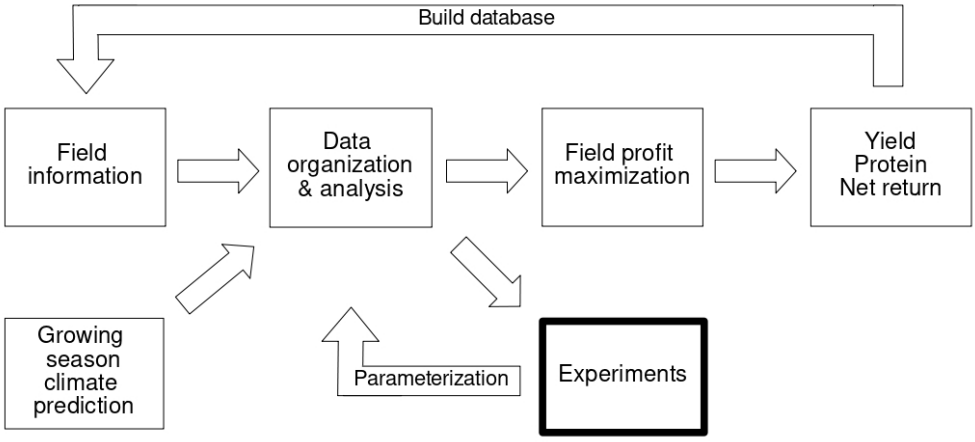


Figure 4.1: Flowchart of the optimization process for field profit maximization.

we have $\sum_{i=2}^{b} \binom{c-((i-2)c/b)}{c/b}$ combinations to consider, where c is the total number of cells and b is the number of nitrogen bins. As the number of cells in a field vary between 120 and 300 cells, we generally have at least 4 nitrogen rates, but there can be as many as 6. Then in the "easiest" case where $c = 120$ and $b = 4$, we have $\binom{120}{30} + \binom{90}{30} + \binom{60}{30} \approx 1.7 \times 10^{28}$ combinations. As we can see, an exhaustive search is impractical. The specific implementation is explained in more detail in Section 4.2.

4.1 Background

GAs are part of the family of Evolutionary Algorithms (EAs). EAs are metaheuristic optimization algorithms inspired by biological evolution. They have been shown to work especially well on multi-objective optimization problems [40]. EAs are not the only metaheuristic optimization methods that find their inspiration in biology. Swarm Intelligence (SI) bases itself on the collective behaviour of self-organized systems. The most well known SI implementations are Ant Colony Optimization [30] and Particle Swarm Optimization [34]. Due to the multi-objective nature of our problem, a GA approach is a natural fit. Furthermore, even though

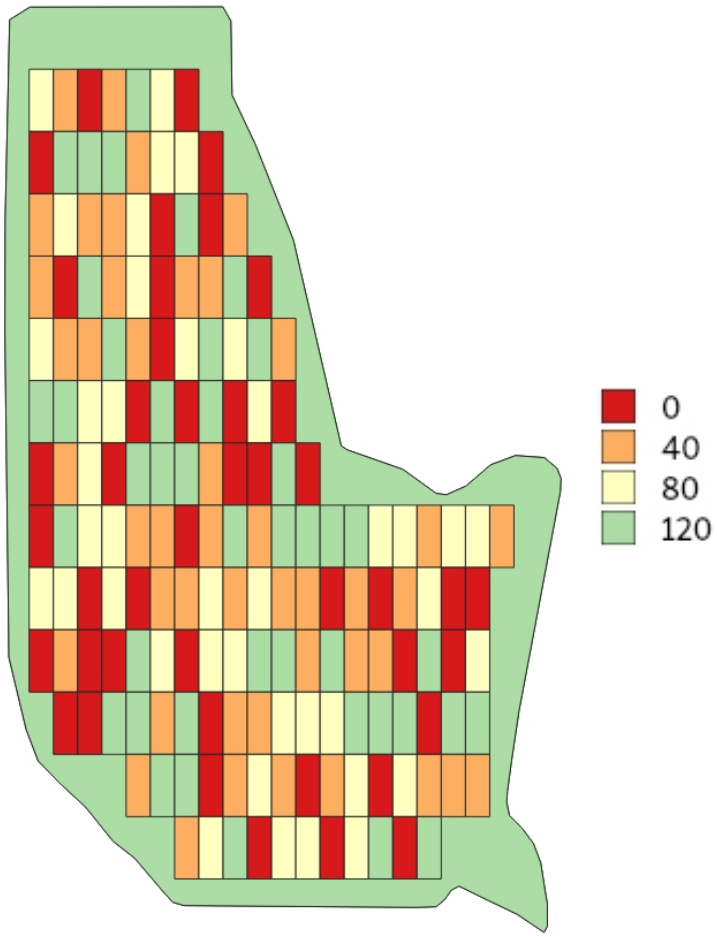


Figure 4.2: Unoptimized prescription map of field davidsonmidwest. Fertilizer levels in terms of pounds of nitrogen per acre are shown in the legend.

(a) Example of consecutive cells with large jumps.

(b) Example of consecutive cells with small jumps.

Figure 4.3: Example of four consecutive cells in a field with large and small jumps. Red, orange, yellow and green correspond to 0, 40, 80 and 120 pounds of nitrogen/ha respectively.

GAs cannot guarantee an optimal solution, they have been shown to work well on real world applications in many different fields [50].

The idea of the GA specifically is motivated by biological evolution and the Darwinian concept of "survival of the fittest" [56]. A population of candidate solutions (in this case prescription maps) is generated and then modified using a stochastic sequence of selection, variation, and replacement. A candidate solution is also called a chromosome, and each chromosome consists of genes. In its simplest form, these genes are represented by binary values; however, the representation can also consist of integers, real numbers, permutations, or trees. The actual values are known as alleles. In our case a candidate solution is a prescription map with grid cells as the genes.

Each chromosome is assigned a fitness value, determined by a fitness function which is chosen to fit the problem at hand [37]. Parents are selected from the current population to create children through crossover and mutation. Parent selection chooses two chromosomes to produce new offspring with. Selection mechanisms favor fitter individuals to be a parent. A method used commonly is tournament selection, which creates a pool of candidate chromosomes by randomly selecting a number of individuals from the population. Out of this candidate pool, the fittest individual is

chosen to be a parent.

After the parents have been selected, they are combined using crossover. This means that the two parent chromosomes are recombined in a specific way to create two new child chromosomes. There are several different ways to perform crossover, a simple implementation is called one-point crossover. A random point in the parents is selected, the part before this point in the first parent is combined with the part after the point in the second parent to create the first child. The second child consists of the opposite elements. A crossover rate is implemented to decide probabilistically whether crossover should be performed. This rate is usually set to be rather high (above 90% chance), as crossover is favorable to generate new individuals. After crossover, mutation is performed to alter the resulting child chromosomes further. Once again, whether mutation is performed or not is decided by a probabilistic rate; however, this rate is set to be relatively low (often below 10%). Mutation only changes a small part of the chromosome. An example of a mutation operator is the flip mutation for binary representations. Each gene is considered separately; the mutation rate decides whether that specific gene's bit value is flipped to the opposite value.

Once the children have been generated the new population has to be created. This can be done in several different ways. The two main methods are generational and steady-state replacement. Generational replacement changes the entire population by replacing it with the generated offspring. Steady-state replacement only changes part of the population, in other words, it keeps part of the old population. For example, the children can simply replace the parents entirely, given the amount of children generated is the same as the population size, or the children and parents can be grouped together and the fittest individuals from the overall group form the new population. The overall workings of a GA can be seen in Figure 4.1.

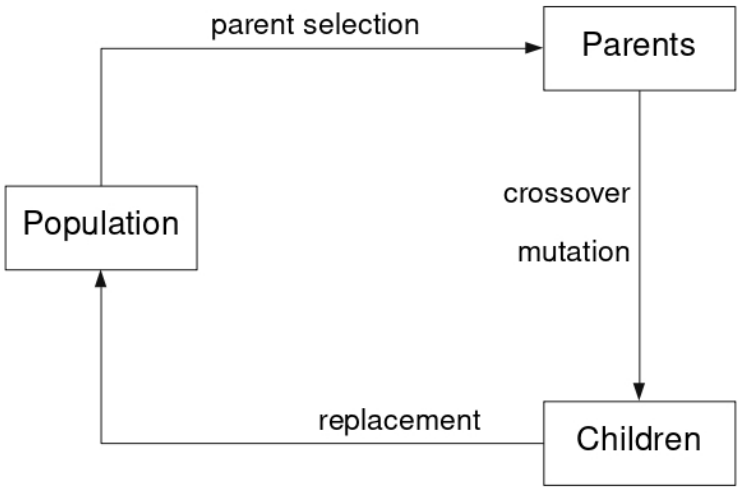


Figure 4.4: The general workflow of a genetic algorithm.

4.2 Methodology

As previously mentioned, one of the problems addressed in this research is the design of a random, stratified experiment for collecting data in precision agriculture. Our goal is to specify experimental prescription maps that maintain stratification over a space of prior yield and protein production in winter wheat while simultaneously minimizing the magnitudes of jumps in nitrogen application over the map. To facilitate measuring jump magnitudes, a previous "as applied" map is used to determine the intended route the the spreader/sprayer is to follow. We hypothesize that we can apply a genetic algorithm to generate experiment prescriptions that maintain stratification effectively and minimize jumps in nitrogen rate application. In this study, we test this hypothesis by considering a variety of genetic operators and by examining the effects of these operators on overall fitness as well as individual impact on stratification and smoothness.

We apply a GA to optimize a fertilizer experiment prescription map, which

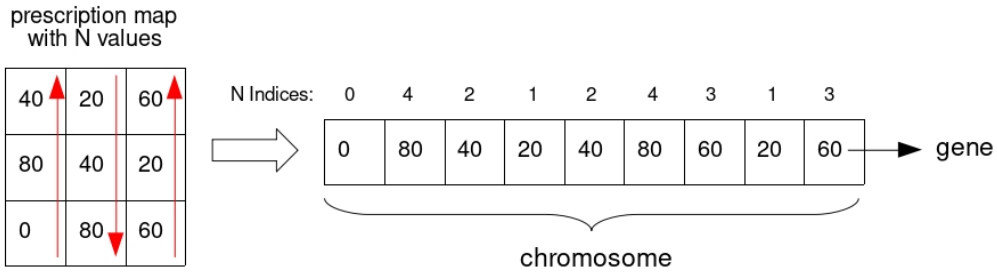


Figure 4.5: Diagram showing how a prescription map is transformed into a chromosome, and how this chromosome is represented using indices.

dictates the nitrogen application rate for each cell on a field. In actual use, the farmer decides which k nitrogen rates they wish to apply and how the field is to be subdivided (Figure 4.2). The ultimate goal is to minimize jumps (Figure 4.3) while maintaining stratification as best as possible.

For our GA, we use completed prescription maps as individual chromosomes in our population with each cell in the map as a gene and its corresponding nitrogen rate as the gene's allele. Each nitrogen rate maps to an index which is used to calculate the jump score (Figure 4.5). A multi-objective fitness function is then applied, taking jumps as well as stratification into account.

4.2.1 Stratification

The stratification strategy tries to ensure that each nitrogen rate is represented evenly in each of the bins. Let $B = \{b_1, \ldots, b_k\}$ denote k total bins, representing the yield-protein combinations, where yield has been divided into q sections and protein has been divided into p sections. Thus, we have a total of $k = q \times p$ bins. And for $b_i \in B$, $C(b_i)$ is the set of cells in b_i. Then $|C(b_i)|$ corresponds to the number of cells in the field that map to a specific bin. We then compute target stratification as

$$tstrat_{bin} = \frac{1}{k}|C(b_i)|$$

since our goal is to distribute the nitrogen evenly over exactly k bins. The initial prescription maps are generated by computing bins based on the field's previous year yield and protein data.

We consider two different methods for discretization into bins: 1) by looking at the actual yield (yd) and protein (pro) values (called equal width binning), or 2) by splitting on the data points themselves such that each bin has the same number of data points (called equal sample binning). The first method looks at the minimum and maximum yield and protein values and creates an even split of these values based on the desired number of bins. Without loss of generality, consider yield. Then based on the number of bins q we calculate an offset as

$$\mathit{offset}_q = \frac{1}{q}(\max_{yd} - \min_{yd}).$$

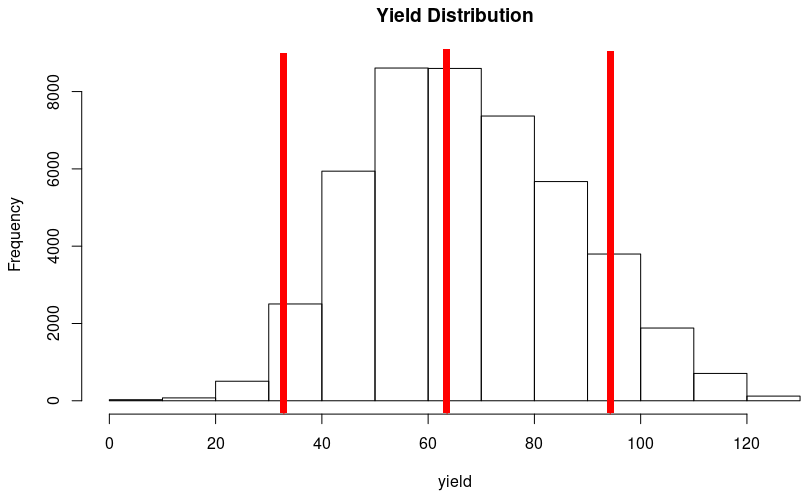
Thus, we get bin boundaries at

$$\min_{yd}, \ldots, \min_{yd} + j \cdot \mathit{offset}_q, \ldots, \max_{yd}.$$

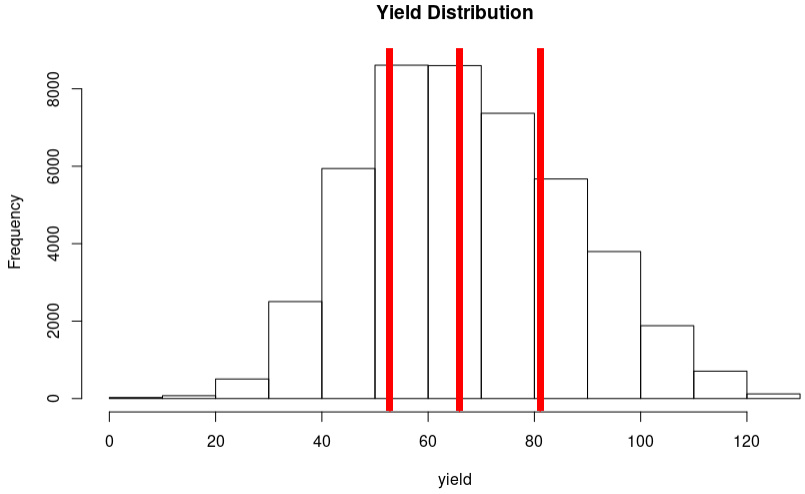
Equal sample binning, where we split on m data points, does not take the yield or protein values into account but aims to distribute an even number of points into each bin (i.e., m/q points for yield bins and m/p points for protein bins). The differences in binning strategies are illustrated in Figure 4.6.

4.2.2 Genetic Algorithm Implementation

The initial population consists of n prescription maps, where each map is a chromosome such that its c cells are genes in the chromosome. Each cell belongs to a bin combination $l \in \{1 \ldots k\}$ and is assigned a nitrogen rate. Let $R = \{r_1, \ldots, r_i\}$ be the set of nitrogen rates. The corresponding nitrogen index i is denoted as



(a) Equal Width Binning



(b) Equal Sample Binning

Figure 4.6: Example of different bin discretization types using a histogram representation of the yield values. The vertical red lines indicate bin boundaries using each discretization type.

$N(map_{cell}) \in \{0 \ldots |R| - 1\}$, for each cell map_{cell} on the field. For example, if the chosen N rates are $\{0, 20, 40, 80\}$, the corresponding indices would be $\{0, 1, 2, 3\}$. Once the population has been generated and evaluated, tournament selection is performed,

choosing a predefined number of pairs by selecting the best map (lowest fitness score) from a chosen number of individuals from the current population. For each of these pairs, two-point crossover is performed by randomly selecting two indices and swapping the cells between these indices to create two new child prescription maps. These new maps then replace the two maps in the original population with the worst fitness score.

Mutation is then applied to the offspring to maintain diversity in the population. Two mutation approaches were implemented. Swap mutation chooses two random indices and switches the values of these two cells, and scramble mutation is performed by selecting all cells between two randomly chosen cells and performing a random permutation. Finally, the λ children replace the λ worst individuals in the population to create the new population, which is a steady-state replacement strategy [36].

There are several parameters that can influence the performance of the GA: the population size, the number of offspring to create, the number of candidates in tournament selection, and the crossover and mutation rates. The mutation (0.05, 0.10, and 0.15) and crossover (0.90, 0.92, 0.95, 0.98) rates were tuned simultaneously, where each combination of the two was tested. Population size (200, 400, and 800), tournament size (2, 3, 5, 10, and 20), and offspring (20, 40, 80, 100) were tuned individually; the best result was used while tuning the other parameters. After tuning, all experiments were run using tournament sizes of 3 and 20 to compare the behavior of the GA in more detail. The final values for each of the hyper parameters are shown in Table 4.1.

Fitness is determined by creating a multi-objective minimization function based on jump and stratification score. The jump score sums over the absolute difference in nitrogen levels between adjacent indices along the as-applied map of nitrogen rates. The as-applied map dictates the path the spreader follows when applying fertilizer

Table 4.1: Chosen values for all hyper parameters. The parameters are population (Pop), offspring created (OS), crossover rate (CR), mutation rate (MR), and tournament size (TS).

Parameter	Pop	OS	CR	MR	TS
Value	400	40	0.9	0.1	3 or 20

on the field, this map is used to determine the order of the cells $\{map_1, \ldots, map_i\}$.

$$\Delta jumps_i = |N(map_i) - N(map_{i+1})|$$

where $N(map_i)$ corresponds to the nitrogen index of cell i. A jump difference less than or equal to 1 for the ith cell is not added into the jump score, as this is the most desirable rate change between cells. Each individual jump score is then normalized to be within a [0,1] range:

$$F_{jumps} = \frac{\sum_{i=1}^{c-1} \Delta jumps_i}{max_{jumps}},$$

where

$max_{jumps} = (r-1) \times (n-1)$ is the normalization factor. The maximum value is based on the worst case scenario where each consecutive cell goes from the minimum to the maximum nitrogen rate or vice versa, as illustrated in Figure 4.3a.

The stratification score looks for an even distribution of nitrogen rates across cells belonging to the same bins:

$$F_{strat} = \frac{\sum_{l=1}^{k} |tstrat_l - astrat_l| - min_{strat}}{max_{strat} - \quad min_{strat}}, \tag{4.1}$$

where $tstrat_l$ is the target stratification and $astrat_l$ is the actual stratification of

the same bin. The max stratification is determined by the worst case scenario. This occurs when each cell has the same nitrogen rate, indicating that each bin only has one nitrogen rate in which it puts all its cells; therefore, every other nitrogen cell count for that bin will be set to 0.

To calculate the max stratification, a matrix is created of dimensionality $r \times k$, where rows represent nitrogen rates and columns represent bins. For example, assume there are 3 nitrogen rates and 3 yield and protein bin combinations (e.g. 3 yield bins and 1 protein bin), this creates a 3×3 matrix. Furthermore, assume there exists a field with 45 total cells. If equal sample distribution is applied, the resulting target stratification is $tstrat = 5$ for each bin, as $45/9 = 5$. Therefore, when every cell in the map is set to the first nitrogen rate, this is the resulting matrix:

$$\begin{bmatrix} 15 & 15 & 15 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

This means that for the first row $r = 1$, each bin's stratification difference can be defined as:

$$\Delta strat_{1k} = r - 1 \times tstrat,$$

and for $r \neq 0$ each bin's stratification difference is:

$$\Delta strat_{rk} = tstrat,$$

which occurs $r - 1$ times (i.e. for every nitrogen rate that is not the default nitrogen rate, in this case $r = 1$). Both these differences occur k times, once for each bin.

Summing these two stratification differences, therefore, results in:

$$\max_{strat} = 2 \times tstrat \times k \times (r - 1).$$

Minimum stratification can be obtained by determining the number of cells remaining after the nitrogen rates have been distributed evenly. Consider the following example, instead of 45 cells as in the previous example, suppose there are 47 cells, 16 in the first two bins and 15 in the third bin. This means that even if all nitrogen rates are distributed evenly, there will be two cells assigned an "extra" nitrogen rate, either the same nitrogen rate or two different nitrogen rates, making perfect stratification impossible. To accurately describe the remaining stratification and how it would be spread across the nitrogen rates, the number of remaining cells is divided by the number of nitrogen rates r for each yield and protein bin combination. To find the total minimum stratification this intermediate minimum stratification has to be calculated and summed for each bin to determine the minimum possible stratification:

$$\min_{strat} = \sum_{i=1}^{q} \sum_{j=1}^{p} (\frac{\#cells_{ij} \bmod r}{r}).$$

Target stratification is calculated by counting the number of cells belonging to each bin $l \in \{0 \dots k\}$. This is done to determine how many cells each nitrogen rate should have for that specific bin.

$$tstrat_l = \frac{\#cells_l}{r}$$

The actual stratification for a yield and protein bin is calculated by counting the number each nitrogen rate occurs in each of the bins. For example, if you again have 45 cells with 3 nitrogen rates and 3 total bins, we know the target stratification is 5

if each of the bins contains 15 cells. Let's say that the first bin of 15 cells has 3 low N rates, 3 medium N rates, and 9 high N rates. The expected stratification is then subtracted from each of these counts (i.e. $|3-5|$, $|3-5|$, and $|9-5|$). The resulting values are summed to calculate the actual stratification for that bin, as shown in Equation 4.2.2.

The final fitness function is a weighted combination of the normalized jump and stratification scores, where $w \in [0, 1]$:

$$F_{map} = (w \times F_{jumps}) + ((1 - w) \times F_{strat}).$$

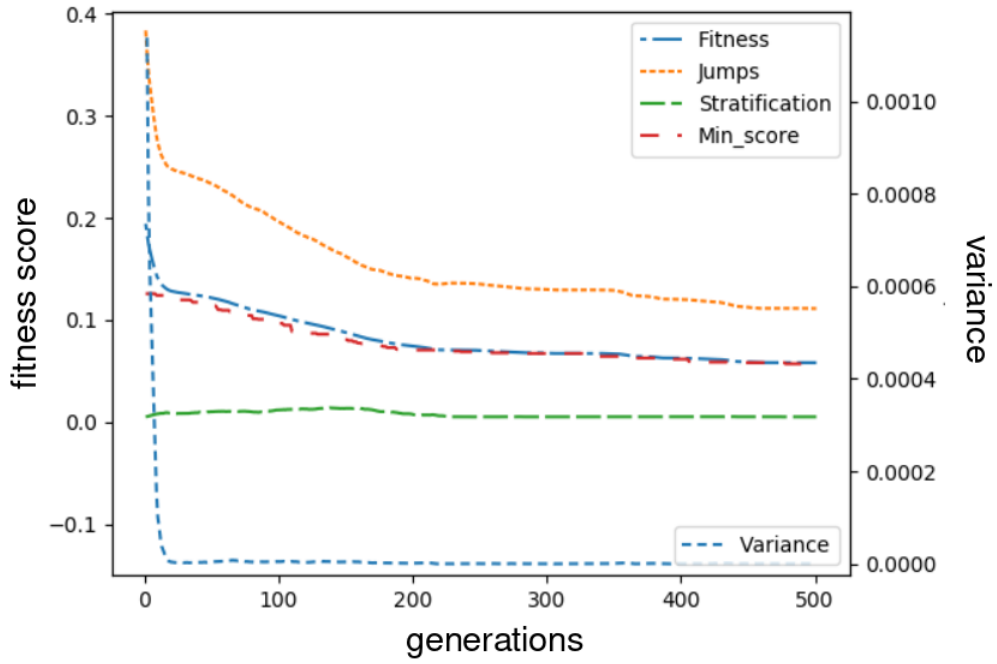
The GA stops running when the set maximum number of generations has been reached. In practice, the algorithm is also set to stop if the farmer deems the jumps to be low enough, the software provides a button to manually stop the GA A.5.

4.3 Results and Discussion

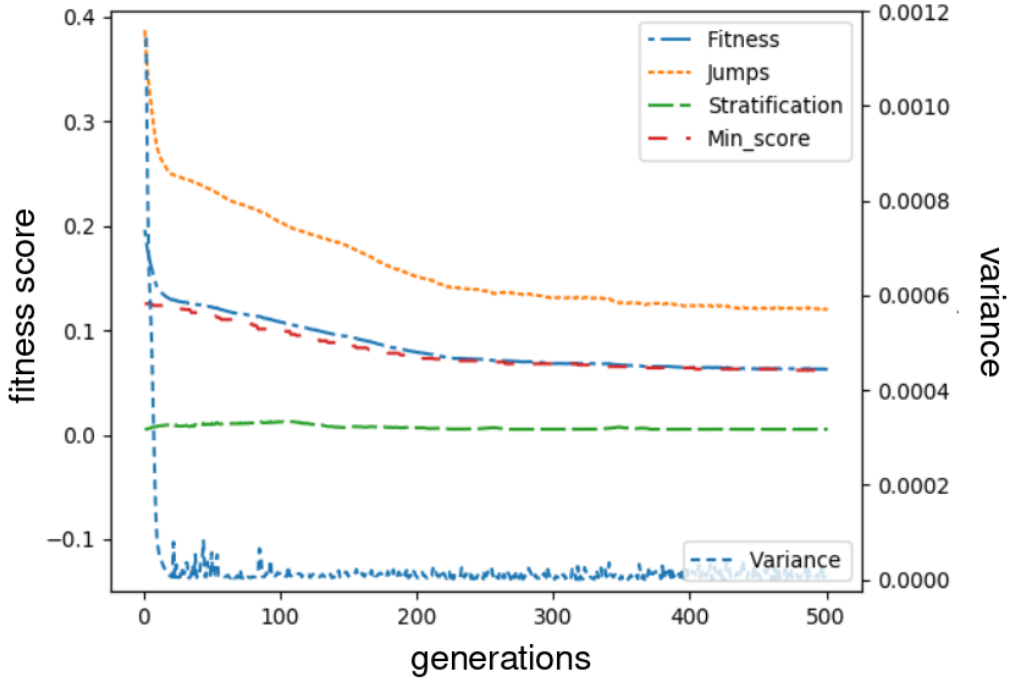
The plots shown in this chapter are for field sre 1314. The plots for the two other fields, sec35mid and davidsonmidwest, can be found in Appendix B.

4.3.1 Equally Weighted Fitness Function

In Figures B.5 and B.6 the jump and stratification scores give equal weight to the final fitness score. The plots show that the GA moves towards convergence, which is the desired result. However, as farmers generally do not have the luxury to wait for hours for a GA to run to find an improved prescription map, i.e. with much lower jumps, increasing the emphasis on the jump score could lead to a more practical use of the GA. Furthermore, using scramble seems to explore more of the search space as there is a slightly larger change in the jump and stratification scores for

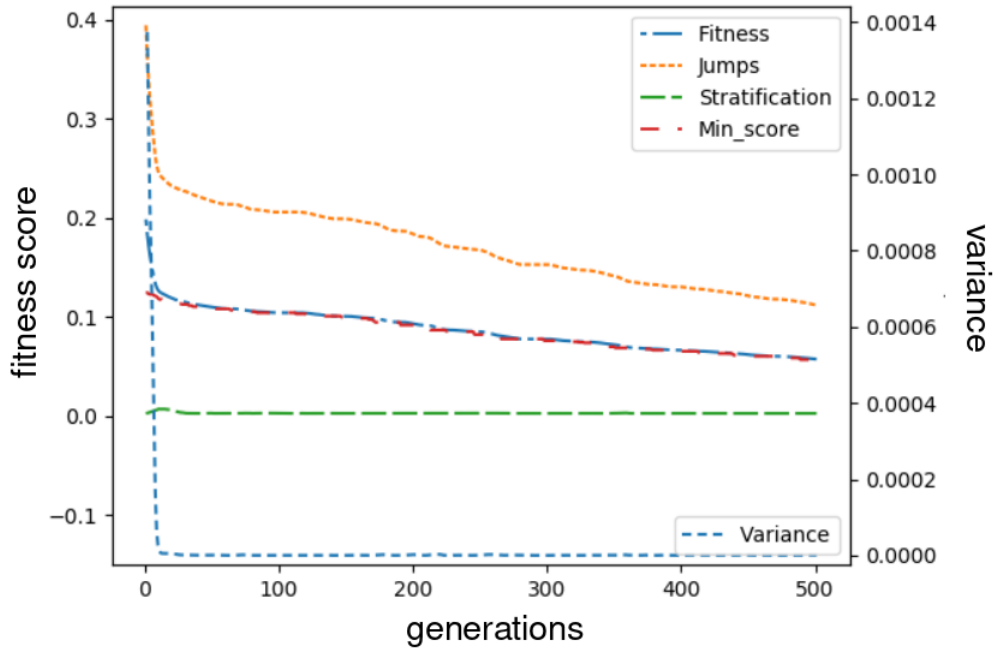


(a) Swap mutation using tournament size 3.

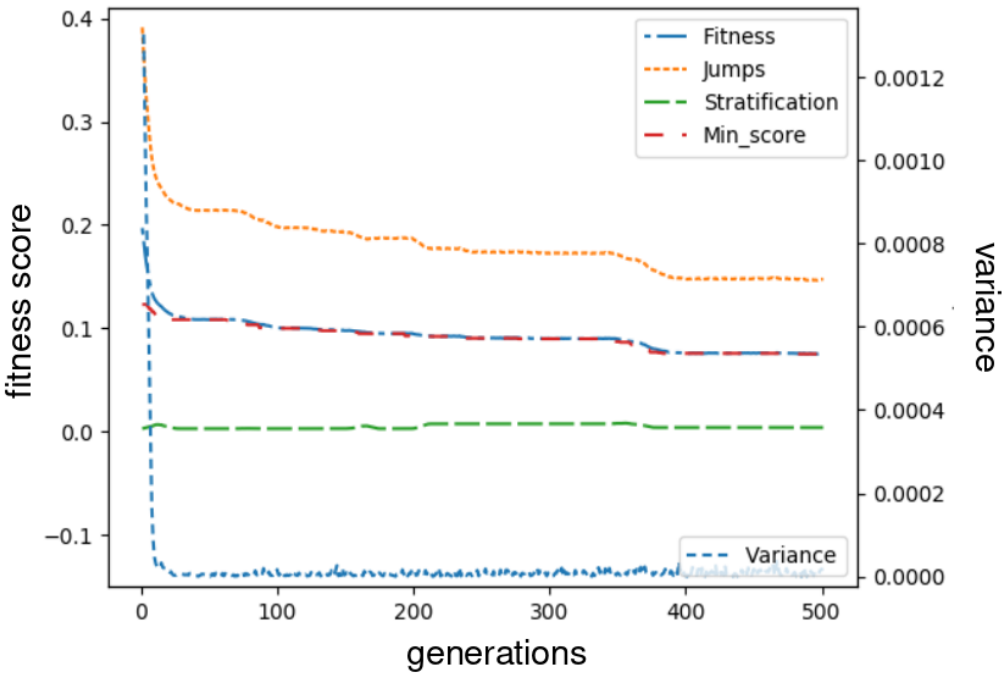


(b) Scramble mutation using tournament size 3.

Figure 4.7: Sre 1314 results for 500 generations of the GA using the two different mutation types and equal sample binning, where $w = 0.5$. The left y-axis shows the fitness score values (including jumps, stratification and the best score in the population), while the right y-axis details the variance value.



(a) Swap mutation using tournament size 20.



(b) Scramble mutation using tournament size 20.

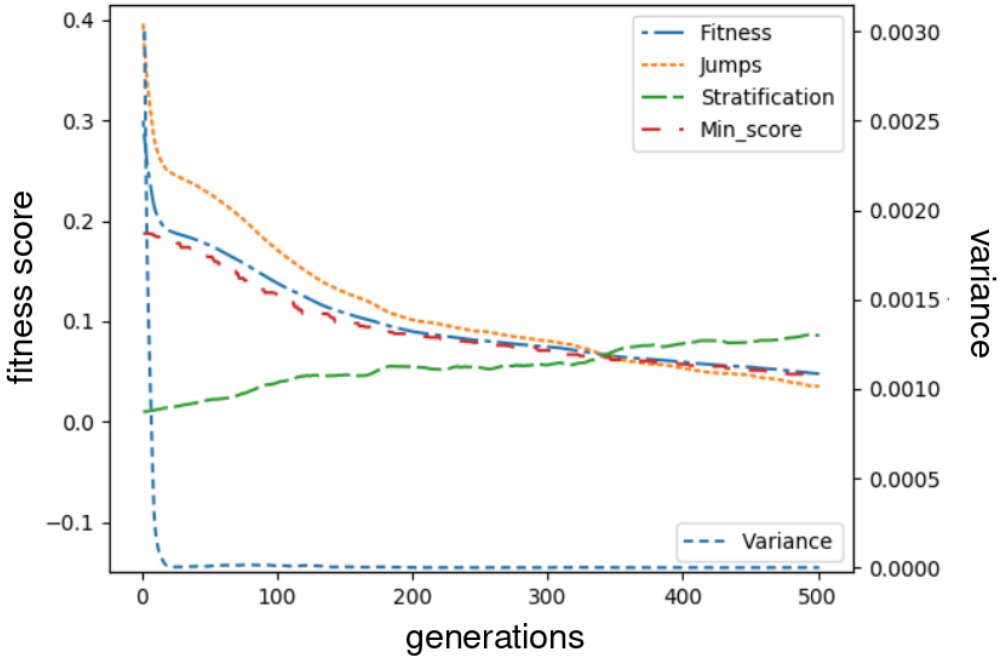
Figure 4.8: Sre 1314 results for 500 generations of the GA using the two different mutation types and equal sample binning, where $w = 0.5$. The left y-axis shows the fitness score values (including jumps, stratification and the best score in the population), while the right y-axis details the variance value.

the population. More importantly, there are much larger changes in variance of the population across the generations. When comparing the behaviour of the GA with a different number of individuals for tournament selection, using a smaller number of individuals seems to reach more rapid convergence than using a larger number. As increasing the number of individuals increases selective pressure, this result seems counter-intuitive.

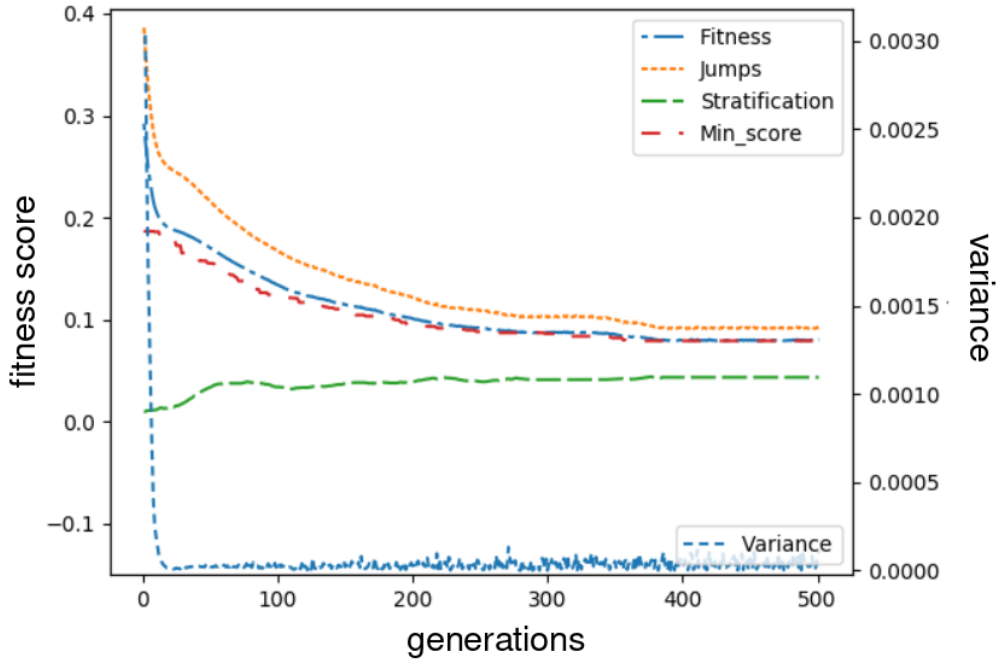
4.3.2 Fitness Function with Focus on Jump Score

Figures B.7 and B.8 show the results for 500 generations of the GA using swap and scramble mutation respectively and using the two different discretization methods, each using a weight of $w = 0.75$. The results again indicate convergence for both the scramble and swap mutation methods. It is interesting to note that the swap mutation seems to find lower jump scores than scramble. This might indicate that scrambling changes the maps too much, producing offspring that do not reduce the jump or stratification score. This would also explain why convergence is slower, as it would take longer to find better prescriptions. Swap mutation, on the other hand, makes smaller adjustments, thereby possibly providing maps that better maintain the overall stratification while exploring a minimization in jumps.

In all cases, there is a substantial drop in variance of the fitness scores early on in the process. The initial variance is small to begin with but becomes almost negligible after a few generations. However, a clear change in variance is evident when applying scramble mutation. This indicates that the population fitness becomes very similar early on. Considering the initial prescriptions are being created with the goal of laying out a randomly stratified prescription map for nitrogen rates based on yield and protein bins, it makes sense that there would not be much variance in the overall fitness. Once the jumps start to drop, the fitness scores would become

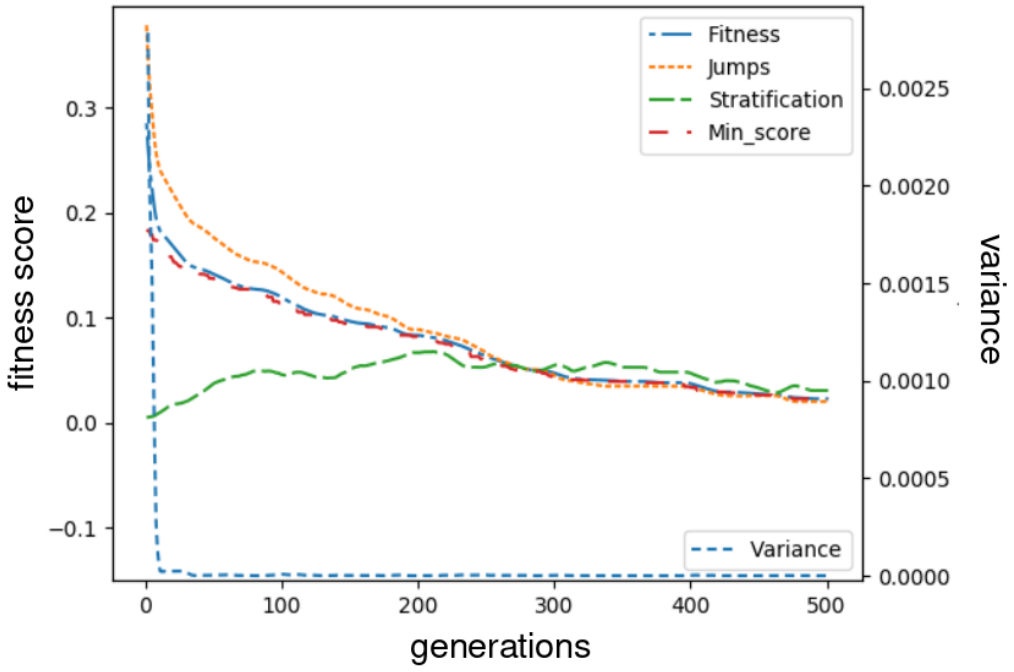


(a) Swap mutation using tournament size 3.

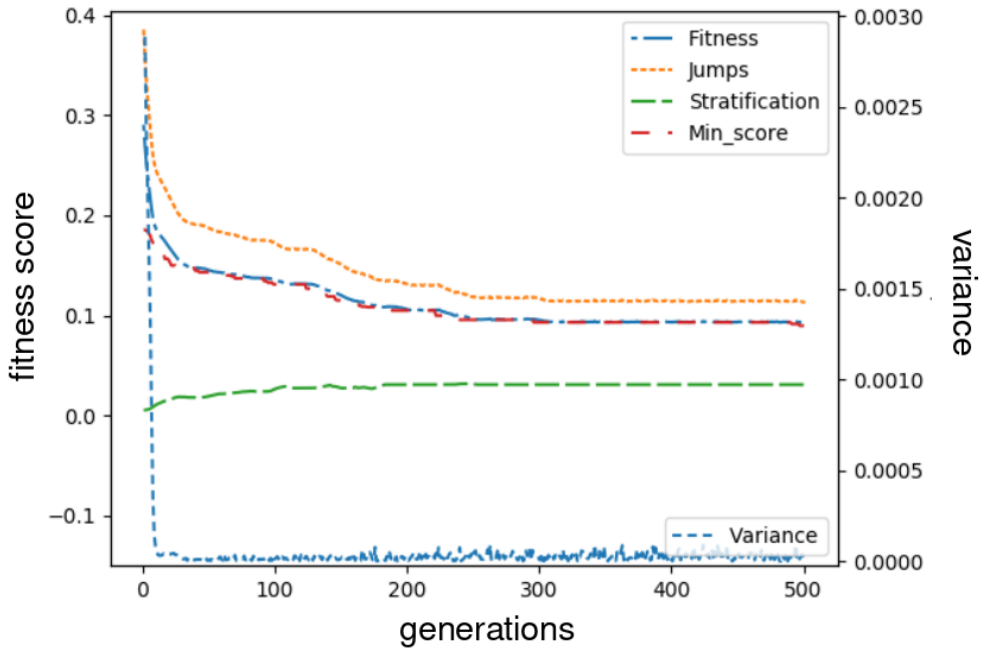


(b) Scramble mutation using tournament size 3.

Figure 4.9: Sre 1314 results for 500 generations of the GA using the two different mutation types and the two different discretization methods, where $w = 0.75$. The left y-axis shows the fitness score values (including jumps, stratification, and the best score in the population), while the right y-axis details the variance value.



(a) Swap mutation using tournament size 20.



(b) Scramble mutation using tournament size 20.

Figure 4.10: Sre 1314 results for 500 generations of the GA using the two different mutation types and the two different discretization methods, where $w = 0.75$. The left y-axis shows the fitness score values (including jumps, stratification, and the best score in the population), while the right y-axis details the variance value.

even more similar. The plots show that the largest drop in jump score also occurs early in the process, making the drop in variance a logical consequence. The resulting fitness scores are rather small because of the way the jump and stratification scores are calculated. The maximum jump and stratification values are much higher than what most prescription maps result in, as these are worst case scenarios. The large difference between the maximum (worst case) and actual scores thus results in a small fitness score, potentially explaining the low change in variance, as well as the small difference between the average and minimum fitness score.

4.3.3 Statistical Analysis for 10 Runs of the GA

The average total fitness score results for 10 runs of the GA are shown in Table 4.2. A paired t-test is performed to confirm that the scores for scramble and swap mutation are statistically different from each other for all fields and both binning methods at the $\alpha = 0.05$ level. The results using 3 and 20 individuals for tournament selection are significantly different at the $\alpha = 0.05$ level. Furthermore, the results show that the fitness scores for swap mutation are consistently lower, and that the GA achieves a lower fitness score for both discretization methods. Based on these results, scores from a single run of the GA for the original prescription maps and the "best" maps, obtained through swap mutation, are shown in Table 4.3. An illustration of an optimized prescription with minimized jumps can be found in Figure 4.11 under the assumption the spreader travels along the north-south axis. This illustrates a map where the jump score decreased substantially, and the stratification score only went up slightly for the final prescription map, which was the desired result.

Table 4.2: Average fitness score of the best maps after 10 runs of the GA for scramble and swap mutation, using equal width and equal sample binning, on three different fields. The jump weight is set to $w = 0.5$.

		sec35mid		**davidsonmidwest**		**sre1314**	
		3	**20**	**3**	**20**	**3**	**20**
Equal Width	**Swap**	0.0282	0.0207	0.0493	0.0195	0.0582	0.0578
	Scramble	0.0520	0.0401	0.0385	0.0601	0.0627	0.0756
Equal Sample	**Swap**	0.0342	0.0309	0.0425	0.0213	0.0679	0.0525
	Scramble	0.0364	0.0468	0.0489	0.0578	0.0613	0.0744

Table 4.3: Fitness, jump, and stratification scores for the initial map and the final map after one run of the GA using equal sample binning. Results are shown for three different fields, two different tournament sizes (3 and 20), and the jump weight is set to $w = 0.5$. All results are for the swap mutation method as this gave the lowest fitness.

	sec35mid				**davidsonmidwest**				**sre1314**			
	3		**20**		**3**		**20**		**3**		**20**	
	First	**Best**	**First**	**Best**	**First**	**Best**	**First**	**Best**	**First**	**Best**	**First**	**Best**
Fitness	0.2457	0.0341	0.2381	0.0308	0.1598	0.425	0.1889	0.0213	0.2492	0.0679	0.1523	0.0538
Jumps	0.4667	0.0435	0.4637	0.0493	0.3139	0.0793	0.3703	0.0370	0.4932	0.1284	0.3018	0.1047
Strat	0.0248	0.0248	0.0124	0.0124	0.0055	0.0055	0.0055	0.0055	0.0052	0.0075	0.0029	0.0028

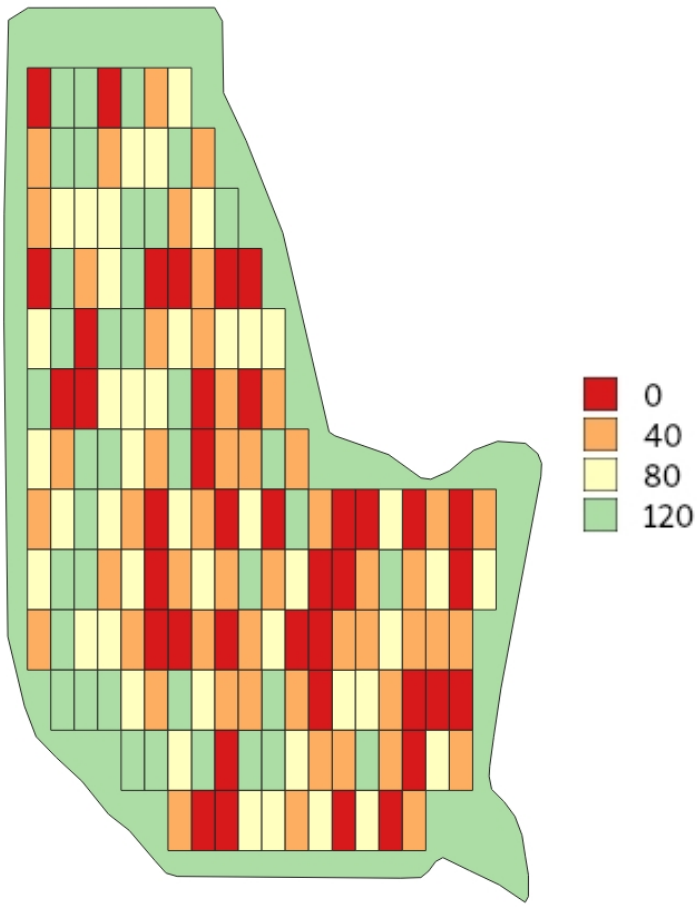


Figure 4.11: Optimized prescription map of the field davidsonmidwest with equal sample discretization and $w = 0.75$. The legend indicates pounds of nitrogen per acre.

CHAPTER FIVE

DEEP LEARNING FOR YIELD AND PROTEIN PREDICTION

As explained in Chapter 2, yield and protein prediction are important factors for farmers growing winter wheat in Montana. This chapter details the initial machine learning approach used to predict the yield and protein content for specific regions in target fields. An overview of the workflow is shown in Figure 5.1, where the stage in which this research falls is bolded. The application of Artificial Neural Networks (ANN), especially within the context of deep learning [46], has been of growing interest for innumerable domains, including image classification, natural language processing, stock market prediction, and medical diagnosis. One of the most interesting advantages of ANNs is their ability to learn and recognize patterns from different input signals, where the network's "neurons" perform the analysis of the input signals simultaneously.

Applying ANNs in the field of Precision Agriculture is a relatively recent development, as was discussed in Chapter 2. Here, we present the results of two different approaches to using ANNs, each within the context of modeling localized properties of the field as well as expanding the inputs to consider spatial context. Specifically, we compare training simple feed forward neural networks to using a deep learning model known as a stacked autoencoder. We then compare the results of these two neural networks to multiple linear regression and multiple nonlinear regression. Our results indicate that the neural network models (both shallow and deep) provide an improvement over the traditional regression methods. Furthermore, we find that adding the spatial context yields a statistically significant boost in performance. The background section discusses each of the models in greater detail. In methodology

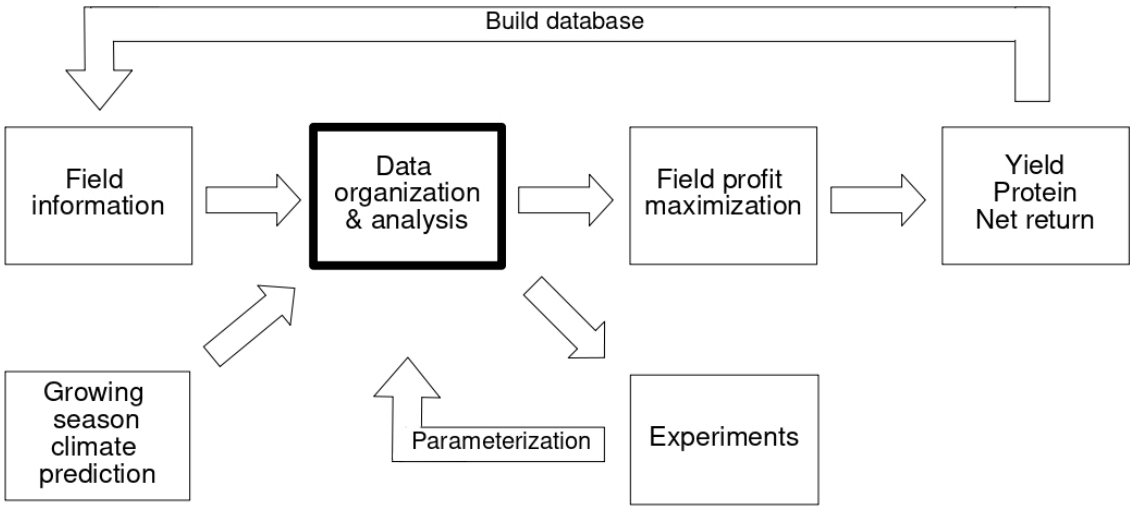


Figure 5.1: Flowchart of the optimization process for field profit maximization. The "Data organization and analysis" stage is highlighted to indicate the main focus of this chapter.

we provide the specifics of our experimental approach, and finally the results of the experiments are presented and discussed.

5.1 Background

An artificial neural network is defined as a parallel, distributed information processing structure, modeled *loosely* on the structure of the brain. This structure contains processing elements connected to each other with unidirectional or bidirectional signal channels (connections). Each of these processing elements possesses local memory and can therefore process local information. All of the processing within an element depends solely on the current value of the input signals and on values stored within the local memory. The elements each have a single output (with an output signal of any mathematical type) branching into as many collateral connections as necessary [53]. This section discusses the feed-forward model in more detail and introduces the concept of a stacked autoencoder as a deep method that,

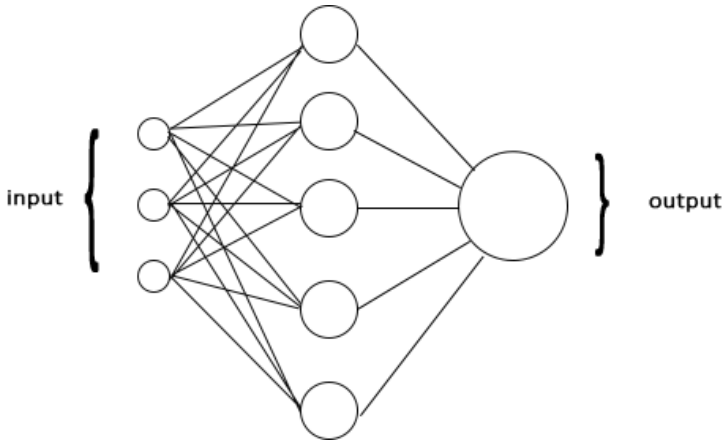


Figure 5.2: A simple feed-forward neural network.

to our knowledge, has not been applied to PA.

5.1.1 Feed-forward Neural Network

In a feed-forward ANN, all nodes of the network pass information forward from the input through a sequence of layers until reaching the output layer. The output of any given node in the network is computed as follows:

$$y = f\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$

where $f(\cdot)$ is often referred to as an "activation function." Letting $z = w_0 + \sum_{i=1}^{n} w_i x_i$, these activation functions take on many forms, including linear ($f(z) = z$), rectified linear (ReLU) ($f(z) = \max\{0, z\}$), logistic ($f(z) = 1/(1 + \exp -z)$), hyperbolic tangent ($f(z) = \tanh z$), radial ($f(\mathbf{z}) = \exp(||\mathbf{z} - \mathbf{c}||/\sigma)$), and softplus ($f(\mathbf{z}) = ln(1 + e^z)$)

The updates to the weights in the interior of the network are defined by the backpropagation (BP) algorithm. An example of a simplified feed-forward ANN is shown in Figure 5.2. BP is one of the most popular techniques for training a neural

network. It was introduced to a large audience by the Parallel Distributed Processing (PDP) group at Stanford University [108]. The PDP lab also developed the first usable technique to implement backpropagation, and because of this they are usually credited for the current approach of BP, even though there were other researchers who discovered BP earlier [16, 129].

In BP, the mean squared error of the network is used as the loss function to be minimized, and the updates correspond to the following:

$$\Delta w_{ji} = \frac{\partial Err(\mathbf{x})}{\partial w_{ji}} = \eta \delta_i x_{ji}$$

where at the output layer,

$$\delta_j = (r_j - y_j) y_j (1 - y_j)$$

using a logistic activation function, with r_j as the target response value, η as the learning rate to adjust how much the weight changes, and x_{ji} as the input from the ith node to the jth node. In the interior layers,

$$\delta_j = o_j (1 - o_j) \sum_{k \in \text{downstream}(j)} \delta_k w_{kj}$$

with o_j being the output generated by the jth node, and downstream indicating the preceding but connected nodes. Note that other activation functions will modify the update rule slightly because of the different derivative being calculated.

A common issue with any type of neural networks is "overfitting" of the data. This occurs when the model is trained using a specific data set, and the hidden nodes have been adjusted to "fit" that specific data too closely, thus not creating a good generalization of the function (Figure 5.3) [69]. This is an important problem to

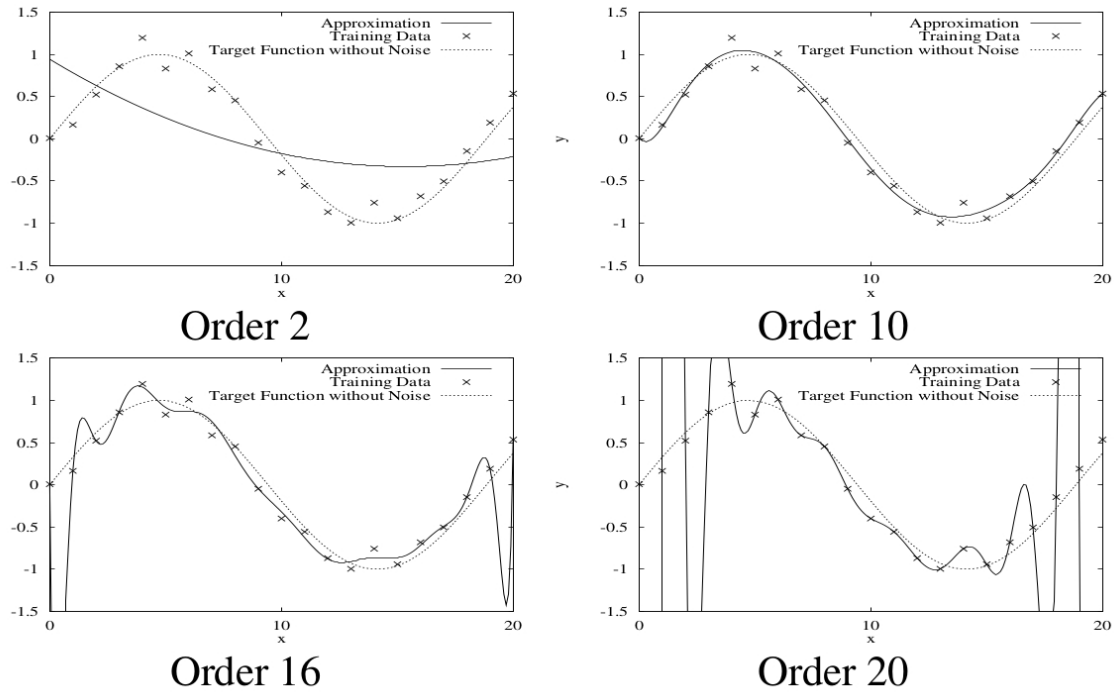


Figure 5.3: An example of overfitting for the function $y = \sin(x/3) + v$ for range 0 to 20 as the order is increased, where v is a uniformly distributed random variable [69].

bear in mind, as good generalization of the function is the overall goal of prediction. Overfitting is especially common with small or sparse data set. Bearing in mind that some of the data we deal with in our research (i.e. the protein data) only has a limited number of points available, this could be a phenomenon to watch out for.

5.1.2 Stacked Autoencoder

The research in this thesis uses a slightly different type of ANN that utilizes deep learning to predict yield, namely the stacked autoencoder (SAE). The main concepts of feeding the data forward and implementing BP to update the network are the same; however the network can be adjusted to use more than two layers effectively. The concept of deep learning arose from the notion that several levels of

abstraction in feature space could be the key to improving the modeling of highly nonlinear relationships among the variables and thus perform better generalizations on complex recognition tasks. This idea has been around since the 1940s with the work of Hebb [52] and McCulloch and Pitts [78]. However, most of the initial networks only consisted of one or two hidden layers. This changed when in 2006 Hinton discovered a new learning approach, the restricted Boltzmann machine, re-energizing research into the field of deep learning.

The autoencoder (AE) was introduced by Bengio *et al.* [9] and Ranzato *et al.* [101] as an ANN to try to copy the input it receives to its output with a hidden layer encoding the input (Figure 5.4). An AE has two functions: 1) encoding the input using the hidden layer, and 2) decoding the encoded values from the hidden layer to reconstruct the input. By design, the AE model is forced to prioritize certain aspects of the input to be copied, thus often learning useful properties of the data [45]. The resulting hidden layer representation can then be used as a condensed representation of the original data, effectively performing feature reduction.

A stacked autoencoder (SAE) is built initially as follows. In the first phase, a single AE is trained using backpropagation with the inputs being treated as target values as well. In the second phase, the first AE's hidden layer is used to create a new AE. In other words, the second AE uses the first AE's hidden layer representation as input to recreate. Because of this, the second network's number of hidden nodes is often smaller than the first's, condensing the data down further. This process is repeated for however many hidden layers the user wants. The AEs are connected by removing the output layer and directly connecting it to the hidden layer from the following AE, which uses the output as input. By repeating this process and connecting the learned AEs, a "stacked" autoencoder results (Figure 5.5).

Since the hidden layers of each AE correspond to a set of feature detectors

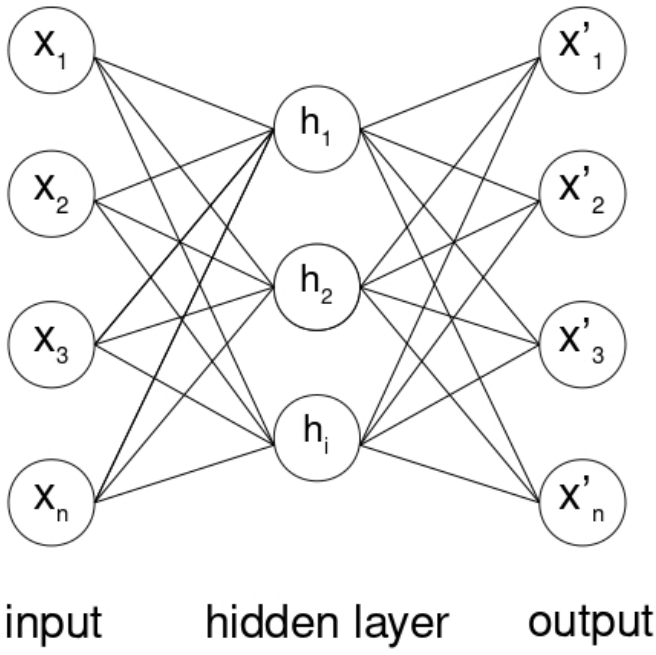


Figure 5.4: A single autoencoder with n features and i hidden nodes.

targeted at the signals from the previous layer, this process of stacking is believed to create an abstraction hierarchy of features for the data being analyzed. Once the AEs have been trained and stacked, the final step is to place another network on the top, focused on the goal of learning (e.g., classification, or in our case, yield or protein prediction). Once this is done, all the weights are trained together to learn the final mapping from the original set of inputs to the target yield/protein values [121].

5.2 Methodology

We implemented simple linear regression through the protein and yield points based on the fertilizer rate. The nonlinear model fit a hyperbolic curve through the data. Using a hyperbolic fit stems from the fact that fertilizer rate hits a saturation point after which yield and protein values no longer rise. We also implemented and

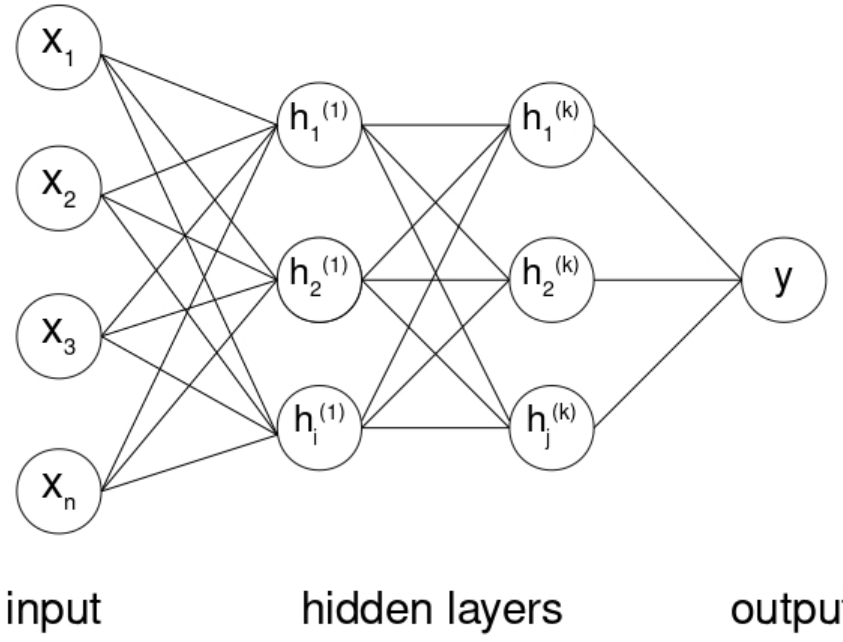


Figure 5.5: A stacked autoencoder with n features and k hidden layers.

evaluated non-spatial and spatial versions of the ANN and the SAE, resulting in a total of six different models. (see Chapter 3 for more information on the linear and non-linear functions as well as the spatial analysis.)

During the experimentation process, different parameters were tested to discover which architecture produced the best results. For the ANN the tuned parameters were the number of hidden layers (1 or 2), the number of hidden nodes (5, 10, 20, 50, 100, and 150). For the SAE the number of hidden layers was tested with 2, 3, and 4 layers. The number of hidden nodes within those layers always decreased in size and several different configurations were tried, always reducing the number of nodes for the consecutive layer. The initial node sizes were chosen at 750, 500, 250, 100, and 50 nodes. Due to the large number of parameters to be tuned for 16 different data sets, the tuning process did not go over all possible combinations, rather a selective

process was used that moved toward the more promising parameter combinations. For example, when tuning the SAE, six experiments would be run initially, starting with 2, 3, and 4 layers and 750 and 250 hidden nodes in the first layer (i.e. the three layer networks would have 750 - 500 - 250 and 250 - 150 - 75 hidden nodes in each layer respectively). Depending on the results the next parameters to test were chosen. These parameters were adjusted for each dataset; however, the ANN achieved optimal performance using a single hidden layer with 5 to 100 hidden nodes, depending on the number of features available for the field. The architecture of the SAE varies between two and three hidden layers for non-spatial and spatial inputs respectively, where the hidden nodes for these layers reduce in size in each layer. For the spatial data, 500, 250 and 125 hidden nodes were used from the top to bottom layer respectively. The number of hidden nodes for non-spatial data was less consistent, depending on the number of data points available.

All of the models were evaluated using Root Mean Squared Error (RMSE), which was calculated using 10-fold cross validation, after which the average of the results for each fold was calculated. A paired Student t-test, adjusted using the Bonferroni correction [11], was performed to evaluate the differences of the means, with a significance value of $p = 0.05$.

5.3 Results and Discussion

Tables 5.1 and 5.2 show the RMSE results for yield and protein predictions respectively. These results are also visualised in Figures 5.6 and 5.7 respectively for a more intuitive overview. The non-linear hyperbolic model did not provide results for protein predictions, which is denoted as "N/A" in Table 5.2. The performed t-tests report p-values lower than 0.05, suggesting the performance of the models is significantly different.

Table 5.1: Yield prediction results for all four fields. Statistically significantly different results are shown in bold. "Sp." stands for spatial data.

Field	**Linear**	**NonLin**	**ANN**	**ANN Sp**	**SAE**	**SAE Sp**
sre1314	11.23	11.25	11.51	11.05	10.89	10.15
sec35mid	10.16	10.01	10.64	**8.97**	10.09	**8.95**
david	15.25	15.92	16.36	**12.49**	16.03	**12.06**
carlin	12.07	10.20	10.23	**8.06**	9.93	**9.15**

Table 5.2: Protein prediction results for all four fields Statistically significantly different results are shown in bold. "Sp" stands for spatial data.

Field	**Linear**	**NonLin**	**ANN**	**ANN Sp**	**SAE**	**SAE Sp**
sre1314	1.22	N/A	**1.16**	**0.90**	1.15	1.13
sec35mid	1.40	N/A	1.44	1.22	1.97	1.96
david	1.68	N/A	**1.60**	**1.49**	1.67	1.61
carlin	1.17	N/A	1.18	1.13	1.37	1.29

The results for yield prediction suggest that spatial data generally improves a model's accuracy (Table 5.1). With the exception of one field (sre1314), the spatial ANN and spatial SAE perform significantly better than the other models. The lack of improvement on this field may be due to the way the yield points are distributed across the field; the points are closer together along the length of the field but are spaced much wider apart across the width than those in other fields. This effectively increases the number of points in one cell of the grid while decreasing the number of total cells to be taken into account for the spatial data. Overall, applying machine learning techniques to spatial data outperforms the more traditional approaches to

yield prediction. More research into other methods of both regression techniques and spatial sampling techniques could provide useful insights into yield prediction and further increase accuracy.

The results for protein prediction tell a different story (Table 5.2). Although improvement is always achieved within the results for the same model using spatial data, the results are not significantly different from each other. However, the ANNs performed significantly better than the other models on two of the fields. Interestingly, the two fields are the ones with the highest and lowest numbers of protein points. More in-depth analysis of the data may provide additional insights into the cause of this phenomenon. Furthermore, the SAE performs relatively poorly, especially when compared to its performance on yield prediction. The most logical explanation is the much lower number of data points available for training the protein models, preventing the models to learn effectively, probably overfitting the data. However, looking at the RMSE values, the predictions themselves are fairly accurate across all models. When analyzing the data, it becomes apparent that protein values do not differ as much as yield values do, possibly making it easier to predict the correct percentage of protein and leading to smaller RMSE values overall.

As a last note, it is difficult to give an accurate representation of the shallow neural networks' performance. When running experiments, the RMSE results suffered from a large amount of variance. This can be seen when comparing the results for the same ANN in this chapter and Chapter 6, meaning that no one run had the same results, and that there was a relatively large amount of spread between these results. This is a well known issue with neural networks, but it can be addressed by applying ensemble methods [14].

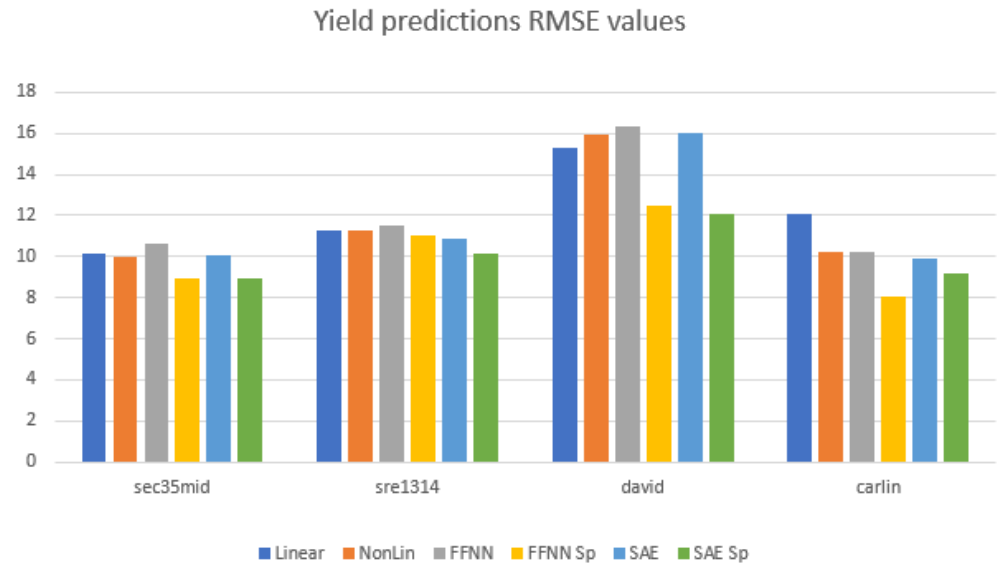


Figure 5.6: Bar chart visualization of the yield RMSE results for the different models.

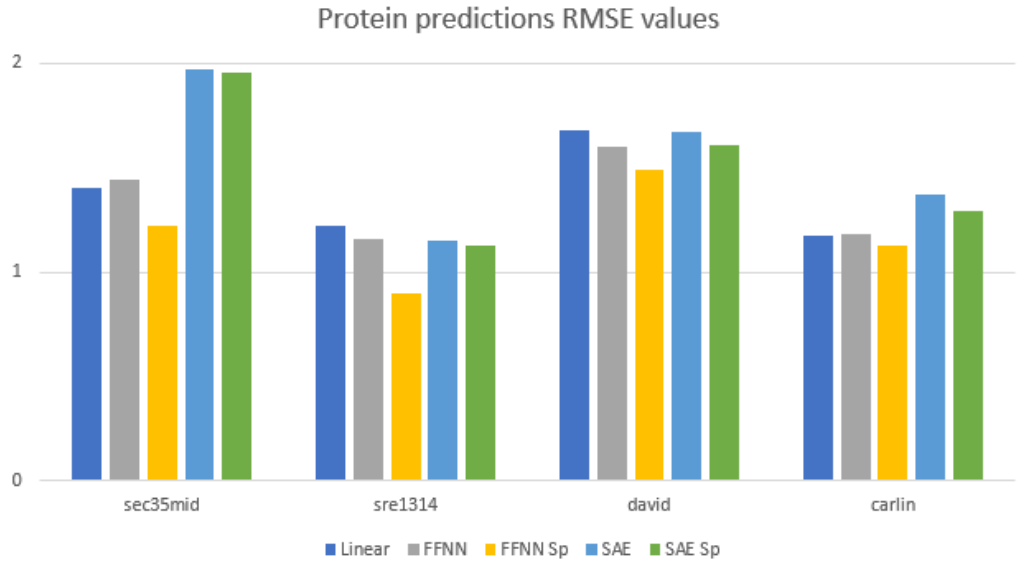


Figure 5.7: Bar chart visualization of the protein RMSE results for the different models.

CHAPTER SIX

ENSEMBLE METHODS FOR YIELD AND PROTEIN PREDICTION

Ensemble methods are a subset of algorithms that combine multiple models. This work implements two different ensemble techniques, bagging and boosting, to predict yield and protein values. This is part of the "Data organization and analysis" step in the project workflow, as can be seen in Figure 6.1. Bootstrap aggregation, or bagging, randomly selects a subset of the original data set. A weak model is then trained on this subset, and the process is repeated until a certain number of models have been trained. These models are then combined to create the final predictor [14].

In this work, we use a specific type of boosting called Adaptive Boosting, or AdaBoost. AdaBoost trains several weak models on the same data but with different sampling weights for each model and applied a weighted combination of the predictions across all models for the final prediction [42]. AdaBoost was first proposed to solve classification problems using decision trees but has since been applied to several different classification and regression problems using different models as the weak learners. In this paper, two existing versions of AdaBoost—AdaBoost.R2 and AdaBoost.RΔ—are applied to a real-world problem in Precision Agriculture (PA). A third novel algorithm combining ideas of these two versions is also applied. All three algorithms use a single layer Feedforward Neural Networks (FFNN) trained with backpropagation [107] as the weak model. Ensembles of FFNNs are not common in PA, and to our knowledge, AdaBoost has not yet been applied to the problem of yield prediction. The goal is to improve yield and protein prediction of winter wheat further using both non-spatial and spatial data.

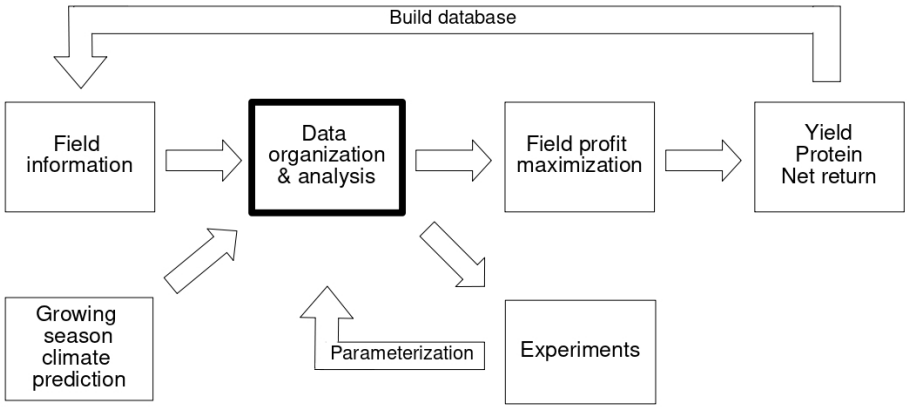


Figure 6.1: Flowchart of the optimization process for field profit maximization.

6.1 Related Work

In our research, both bagging and boosting use neural networks as the weak model. Due to the popularity of neural networks, there have been numerous research efforts in these areas; however, these specific methods have not been applied to precision agriculture. Therefore, this section looks at some successful applications of bagging and boosting implementations with neural networks as the weak model in other fields.

6.1.1 Bagging Neural Networks

Bagging is a fairly simple way to improve performance of unstable methods such as neural networks [14]. This section lists different areas of research where bagging has improved a neural network's performance. Moretti *et al.* [82] predicted one hour urban traffic flow rates using a FFNN, as well as a basic ensemble method (BEM) —which simply averages several models without bootstrapping— and a bagging implementation of FFNNs. The research showed that the bagging approach gave the most accurate predictions. In order to determine how likely a customer is to

respond to a product offering based on that customer's purchase history. Ha *et al.* [49] applied logistic regression, which is standard practice for this specific problem, a single layer multi-layer perceptron (MLP), and a bagging ensemble of MLPs. Interestingly, logistic regression has the highest accuracy, but simultaneously had the highest miss rate, resulting in loss of business. The bagged MLP had the most stable performance, and it also had the best "worst" results. Cunningham *et al.* [23] specifically addressed the unstable nature of ANNs and argued in favor of k-fold cross validation as well as bagging to counteract this instability. They apply the two models to the problem of predicting the outcome of In-Vitro fertilization treatment. The bagged networks achieved higher accuracy than the single model approach. Furthermore, bagging also had lower variance, showing it is likely to be more stable. As a last example, Kim and Kang [63] looked at both bagging and boosting of neural networks for bankruptcy prediction. They found that bagging and boosting both significantly improved results, but with no real difference in performance between the two.

6.1.2 AdaBoost and Neural Networks

Using neural networks as the weak learner in AdaBoost has become more prevalent over the past years. In [115], Schwenk and Bengio apply AdaBoost to classifier ANNs for character recognition, showing the decrease in error rates for boosted NNs. In later work, Schwenk and Bengio compare three different sampling weight update methods for AdaBoost, also using ANNs as the weak model [116]. In resampling, a new sub-sample is chosen randomly from the original data set using the calculated probability distribution of the data points. The first method samples a subset once before training the neural network while not updating the sample set through the training process; the second approach uses a different training set for each epoch; and the third approach directly weighs the cost function. The results

indicated that applying AdaBoost has less of an impact on data sets needing larger networks, and all three sampling methods performed similarly.

Liu *et al.* [73] compared four neural network models with different training methods for wind-speed prediction. The four training methods were Gradient Descent and Gradient Descent with Momentum with Adaptive Learning Rate Back Propagation (GD-ALR-BP and GDM-ALR-BP respectively), Conjugate Gradient Back Propagation with Fletcher-Reeves Updates (CG-BP-FR), and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. They applied AdaBoost using the four models as the weak learners and compare the single model results to the ensemble results. Applying AdaBoost improves results for all four training methods, where the CG-BP-FR training method had the best performance overall. Zhou *et al.* [137] suggested that it could be more beneficial to combine many of the weak models as opposed to all the models. In other words, if 20 neural networks are trained, it could improve results if only 19 out of 20 models are combined for the final predictions, where the excluded model satisfies certain constraints. For this purpose, the authors proposed a new ensemble method—GASEN (Genetic Algorithm based Selective ENsemble)—that selects a subset of networks and compares it to bagging and boosting methods. Their results showed that ensemble methods outperform a single neural network; however, GASEN did not always perform better than the other ensemble methods.

6.2 Background

This section explains the four different ensemble methods that were implemented. Bagging is explained first, followed by a brief history of AdaBoost and its different variations. A detailed explanation of the two implemented versions, AdaBoost.R2 and AdaBoost.RΔ, is given. Lastly, the novel approach of Approximate

AdaBoost (AdaBoost.App) is presented.

6.2.1 Bagging

Bagging is a simple ensemble method proposed by Breiman in 1996 [14]. Improvement of a weak learner will occur when it is an unstable procedure. This means that if a small change is made to the training set, this can result in a large change in the predictive model [15]. An example of a stable procedure is k-nearest neighbor, where examples of unstable procedures include linear regression, decision trees, and neural networks.

Bootstrapping is an important part of the bagging algorithm. It draws samples with replacement from a larger data set to create a new data set. By using a replacement strategy, the resulting set can have the same data point multiple times [35]. Bagging (Algorithm 6.1) uses bootstrapping to create a new data set S from the original [14]. This new data set can be the same size as the original or smaller. A ratio r determines the size of S. A weak model is trained on S and the model is stored. This bootstrapping and training process is repeated until the desired number of models has been reached. All weak models are then combined, or aggregated, to determine the final prediction. For classification, this means performing a majority vote; regression requires a different approach, where averaging the models is a common strategy in this case.

6.2.2 AdaBoost for Regression

AdaBoost was first introduced by Freund and Schapire in 1996 for binary classification [42]. The basic algorithm takes a data set with N d-dimensional samples, $\{x_i \in X^d\}_{i=1}^N$, and corresponding labels $y_i \in \{0, 1\}$ as input, as well as the number of models T to be learned. The initial weight vector or probability distribution $\mathbf{p}^0 = [\frac{1}{N}, ..., \frac{1}{N}]^\top$ is set to a uniform distribution across all points. For each weak

Algorithm 6.1: Bagging

1. Input:
   - Dataset $x_i \in X$, $i = 1, 2, \ldots, N$
   - Number of models T
   - Ratio r determining the size of the subset
2. For $t = 1$ to T:
   (a) Sample with replacement a data set S from X of size $N * r$.
   (b) Fit a weak learner $h_t(X)$ to S.
3. Output:
$$H(x) = \sum_{t=1}^{T} h_t(x)$$

learner, the weight vector is provided to select a training set based on the distribution or to assign a weight to each data point as the model is being trained. A hypothesis h_t is returned after training and is then used to calculate the model's loss ϵ_t, which sums the probabilities of the missclassified points. The weight of the model β is then calculated based on the error as follows:

$$\beta_t = \epsilon_t/(1 - \epsilon_t)$$

However, if $\epsilon_t > 0.5$ the iteration is aborted; otherwise, the weight of the model is used to update the weights:

$$p_i^{t+1} = p_i^t \beta_t^{1-|h_t(x_i)-y_i|},$$

where the distribution is normalized using a normalization factor Z. The final hypothesis is a weighted majority vote of all weak learners' hypotheses based on the β value (Algorithm 6.2).

Algorithm 6.2: AdaBoost.M1

1. Input:
   - Dataset $x_i \in X$, $i = 1, 2, \ldots, N$, with class labels $y_i \in Y$
   - Number of models T
2. Initialize the probability distribution $p_i \in D$ where $p_i = 1/N$, $i = 1, 2, \ldots, N$
3. For $t = 1$ to T:
   (a) Fit a weak learner $h_t(X)$ to the training data using weight distribution D.
   (b) Compute error:
   $$\epsilon_t = \sum_{i:h_t(x_i)\neq y_i} D(t)_i.$$
   If $\epsilon_t > 1/2$, abort and set $T = t - 1$
   (c) Compute $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
   (d) Set
   $$p_i^{t+1} = \frac{p_i^t}{Z_t} \times \begin{cases} \beta_t, & \text{if } h_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases},$$
   where Z_t is the normalization factor
4. Output:
   $$H(x) = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$$

Basic AdaBoost was improved and adapted for regression problems by the same authors in 1997 [41]. Their first regression adaptation is called AdaBoost.R and reduces the regression problem to binary classification as follows,

$$h^c(x, y') = \begin{cases} 0, & \text{if } y' < y \\ 1, & \text{otherwise} \end{cases},$$

where y' is the predicted value from $h(x)$. The weak models are combined to determine the final hypothesis by calculating the weighted median based on the β value of each model [41]. Classification in the context of AdaBoost for regression should be

interpreted to mean that a point's error ϵ_i falls above or below a specified threshold. While some AdaBoost regression models do use such a threshold to create a binary classification (e.g. AdaBoost.R and AdaBoost.RΔ) and ignore the points' errors for further calculations, there are also variations that do use ϵ_i to determine model error and update the probability distribution.

Drucker proposed a new adaptation for regression called AdaBoost.R2 (Algorithm 6.3) [32]. The hypothesis is changed to be able to use any loss function as long as $L \in [0, 1]$. This is accomplished by using the absolute difference between the predicted and actual value to calculate the loss function, as opposed to reducing the regression problem to a binary classification. The loss for each training sample is calculated using three different candidate loss functions (linear, square, and exponential), and the overall model loss is the average of each point's loss L_i times its probability p_i. The weight is then updated as:

$$p_i^{t+1} = p_i^t \beta_t^{1-L_i}.$$

The final hypothesis is still obtained by calculating the weighted median of all model hypotheses.

AdaBoost.RT was proposed in 2004 by Solomatine and Shrestha [118] and uses a threshold for prediction accuracy. If the loss or error rate goes above 0.5, the iteration is no longer aborted but continues until the set number of weak learners has been trained. Instead of calculating the loss, AdaBoost.RT calculates the error of the model ϵ_t by summing probability distribution $D_t(i)$ for all points, as long as the error of the point ϵ_i is larger than the provided threshold δ. Where ϵ_i is the relative absolute error:

$$\epsilon_i = \left| \frac{h_t(x_i) - y_i}{y_i} \right|.$$

Algorithm 6.3: AdaBoost.R2

1. Input:
   - Dataset $x_i \in X$, $i = 1, 2, \ldots, N$
   - Number of models T
2. Initialize the probability distribution $p_i \in D$ where $p_i = 1/N$, $i = 1, 2, \ldots, N$
3. For $t = 1$ to T:
   (a) Fit a weak learner $h_t(X)$ to the training data using weight distribution D.
   (b) Compute loss
   $$L_t = \frac{\sum_{i=1}^{N} L_i}{N}.$$
   where,
   $$\text{Linear: } L_i = \frac{|h_t(x_i) - y_i)|}{\sup(h_t : x \longrightarrow y)}$$
   $$\text{Square: } L_i = \left(\frac{|h_t(x_i) - y_i)|}{\sup(h_t : x \longrightarrow y)}\right)^2$$
   $$\text{Exponential: } L_i = 1 - \exp\left(-\frac{|h_t(x_i) - y_i)|}{\sup(h_t : x \longrightarrow y)}\right)$$
   where,
   $$h_t : x \longrightarrow y = |h_t(x_i) - y_i| \text{ for } i = 1, ..., N.$$
   (c) Compute $\beta_t = L_t/(1 - L_t)$.
   (d) Set
   $$p_i^{t+1} = \frac{p_i^t}{Z_t} \times \beta_t^{(1-L_i)}$$
   where Z_t is the normalization factor
4. Output:
   $$H(x) = \inf\left[y \in Y : \sum_{t:h_t \leq y} \log(\frac{1}{\beta_t}) \geq \frac{1}{2}\sum_t \log(\frac{1}{\beta_t})\right]$$

The probability distribution is then updated in the same way as the original AdaBoost for classification but where the model weight $\beta_t = \epsilon_t^2$, and a point is "classified" as correct if its error is below or equal to the threshold value. If the difference is above the threshold, the weight remains the same. This means that the weight of points that

fall within the threshold gets reduced, whereas points that fall outside the threshold remain the same. The final hypothesis is the result of computing the weighted average as opposed to the weighted median.

An obstacle to using AdaBoost.RT is that it requires the threshold value to be chosen correctly for the best performance. Zhang *et al.* [135] developed a method to infer the threshold parameter for each weak learner by looking at the scaled standard deviation of the approximation errors.

A further adaptation known as AdaBoost.RΔ was proposed by Bertoni *et al.* [10]. AdaBoost.RΔ transforms the regression problem into a binary classification of 0 or 1 based on a threshold value (if the difference between the actual and predicted value is larger than a certain threshold, the point is classified as incorrect). Then the binary outcome is multiplied with the corresponding probability p_i, and sums these to obtain the model error ϵ_t. That is,

$$\epsilon_t = \sum_{i=1}^{N} p_i \times HS(||h_t(x_i) - y_i|| - \delta)$$

where,

$$HS(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases},$$

and $||\cdot||$ denotes a norm. The full implementation can be found in Algorithm 6.4.

AdaBoost.RΔ calculates the final hypothesis as

$$H(x) = \underset{y \in [0,1]^m}{\arg\max} \sum_{y} \alpha_t HS(\delta - ||h_t(x_i) - y_i||)$$

where $\alpha_t = \log \frac{1}{\beta_t}$. But for calculating predictions on a test set, this hypothesis calculation is not possible, as the actual value y_i is not known. Therefore, we calculate

Algorithm 6.4: AdaBoost.RΔ

1. Input:
   - Dataset $x_i \in X$, $i = 1, 2, \ldots, N$
   - Number of models T
   - Threshold δ
2. Initialize the probability distribution $p_i \in D$ where $p_i = 1/N$, $i = 1, 2, \ldots, N$
3. For $t = 1$ to T:
   (a) Fit a weak learner $h_t(X)$ to the training data using probability distribution D.
   (b) Compute
   $$\epsilon_t = \sum_{i=1}^{N} p_i \times HS(||h_t(x_i) - y_i|| - \delta)$$
   where,
   $$HS(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$
   If
   $$\epsilon_t > 0.5$$
   $T = t - 1$ and abort loop.
   (c) Compute $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
   (d) Set
   $$p_i^{t+1} = \frac{p_i^t}{Z_t} \times \beta^{1-HS(||h_t(x_i)-y_i||-\delta)}$$
   where Z_t is the normalization factor
4. Output:
   $$H(x) = \sum_{t=1}^{T} \beta_t h_t(x)$$

the weighted average to determine the final hypothesis.

6.2.3 Approximate AdaBoost

Here, we propose a new approach to applying AdaBoost to ANNs for regression, which we call "Approximate AdaBoost" or AdaBoost.App (Algorithm 6.5). The idea behind AdaBoost.App is to combine the loss function error calculation from AdaBoost.R2 with the threshold approach from AdaBoost.RΔ and AdaBoost.RT.

Theoretically, when a data point falls within the threshold, the weight would be set to 0. The biggest difference lies in the error calculation and weight updates. The error of a single data point ϵ_i is calculated as:

$$\epsilon_i = \begin{cases} |h_t(x_i) - y_i|, & \text{if } |h_t(x_i) - y_i| > \delta \\ 0, & \text{otherwise} \end{cases}.$$

The final errors are normalized using a normalization factor Z. The model error ϵ_t is calculated in the same way as AdaBoost.R2 by summing the errors and dividing by the length, and $\beta_t = \epsilon_t/(1 - \epsilon_t)$. The probabilities p_i are updated based on ϵ_i for $i = 1, 2, \ldots, N$ as follows:

$$p_i^{t+1} = \begin{cases} p_i^t \times \beta^{-\epsilon_i}, & \text{if } \epsilon_i > 0 \\ 0, & \text{otherwise} \end{cases}.$$

This means the probability of a correctly classified point, i.e. $\epsilon_i \leq \delta$, is set to 0 instead of staying the same (AdaBoost.RΔ) or being reduced (AdaBoost.RT), and wrongly classified points, i.e. $\epsilon_i > \delta$, have an increased probability rate. While AdaBoost.RT maintains the weight of wrongly classified points and updates the weights of the correctly classified points, AdaBoost.App sets the probability of a correctly classified point to 0 and changes the weight of incorrectly classified points based on how far off they are. The intuition behind this approach is that it is more important to emphasize points that are further away from the target value. Unfortunately, the implemented neural network is unable to train on a sparsely weighted data set, thus the weights are set to 10^{-6} when classified correctly, to minimize their influence effectively. The final model is then a weighted average over the weak models.

Algorithm 6.5: AdaBoost.App

1. Input:
   - Dataset $x_i \in X$, $i = 1, 2, \ldots, N$
   - Number of models T
   - Threshold δ
2. Initialize the probability distribution $p_i \in D$ where $p_i = 1/N$, $i = 1, 2, \ldots, N$
3. For $t = 1$ to T:
   (a) Fit a weak learner $h_t(X)$ to the training data using weight distribution D.
   (b) Compute
   $$\epsilon_t = \frac{\sum_{i=1}^{N} \epsilon_i}{N}.$$
   where,
   $$\epsilon_i = \begin{cases} \frac{|h_t(x_i) - y_i|}{Z_t}, & \text{if } |h_t(x_i) - y_i| > \delta \\ 0, & \text{otherwise} \end{cases},$$
   where Z_t is the normalization factor ensuring $\epsilon_i \in [0, 1]$.
   (c) Compute $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
   (d) Set
   $$p_i^{t+1} = \begin{cases} p_i^t \times \beta^{-\epsilon_i}, & \text{if } \epsilon_i > 0 \\ 0, & \text{otherwise} \end{cases}.$$
4. Output:
   $$H(x) = \sum_{t=1}^{T} \beta_t h_t(x)$$

6.3 Methodology

To determine the parameters of the neural networks, a limited grid search was performed with 10 fold cross validation for each parameter combination. Each neural network uses a single hidden layer, as this is sufficient for universal approximation of non-linear functions. Three different parameters were tuned: number of epochs (200, 300, 400, and 500), gradient descent optimizer (Adam and Root Mean Square Propagation (RMSProp)), and numbers of hidden nodes (5, 10, 20, 50, 75, 100). As

Table 6.1: Chosen number of hidden nodes (one hidden layer) for each field.

	sec35mid		**sre1314**		**davidson**		**carlin**	
	Yield	**Protein**	**Yield**	**Protein**	**Yield**	**Protein**	**Yield**	**Protein**
Spatial	100	10	10	5	50	5	5	100
Non-Spatial	50	50	10	50	75	5	5	5

both Adam and RMSProp adapt the learning rate during training, extensive tuning of the learning rate was less important. We therefore chose to set the learning rate to 0.01. We applied RMSProp over 400 epochs across all data sets. The best number of hidden nodes varied between 5, 10, 50, 75, and 100 as shown in Table 6.1.

Bagging and all three AdaBoost methods were run on each of the four fields' yield and protein data sets, both spatial and non-spatial, with the number of models ranging from 1 to 20. Two examples of such runs are shown in Figure 6.2. A full reporting of all results can be found in Appendix C. These results indicate that increasing the number of models too far tends to diminish the predictive quality for AdaBoost for this specific problem. However, bagging generally seems to perform better with more models. Weight sampling was performed by assigning a weight to each data point as the neural network was being trained, directly influencing the loss function. This method was chosen because it is closest to the original AdaBoost algorithm and was shown by Schwenk and Bengio to perform well given sufficient epochs [116]. The weak learners use mean squared error (MSE) as the loss function for training. The threshold δ was set to 1.5 for AdaBoost.RΔ and AdaBoost.App. Each of the algorithms was run using 10-fold cross validation, and the results for the test predictions were evaluated using the root mean squared error (RMSE) on the unnormalized values. The RMSE values averaged over the ten folds were then

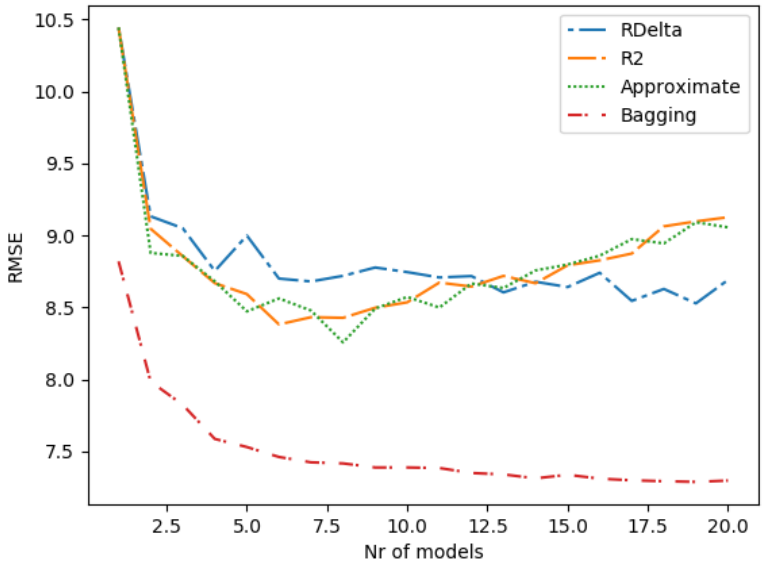
Table 6.2: Yield prediction RMSE results for bagging using four different subset selection ratios: 0.2, 0.4, 0.6, and 0.8. Each of the ratios was run with 20 weak models.

	sre1314		**sec35mid**		**davidson**		**carlin**	
	spatial	**non-spatial**	**spatial**	**non-spatial**	**spatial**	**non-spatial**	**spatial**	**non-spatial**
0.2	9.227	10.316	4.856	7.297	7.546	13.937	5.770	175.94
0.4	9.154	10.235	4.408	7.001	7.088	13.831	5.345	96.45
0.6	9.129	10.205	4.220	6.934	6.882	13.592	5.317	128.29
0.8	9.102	10.174	4.167	6.829	6.847	13.563	5.267	72.11

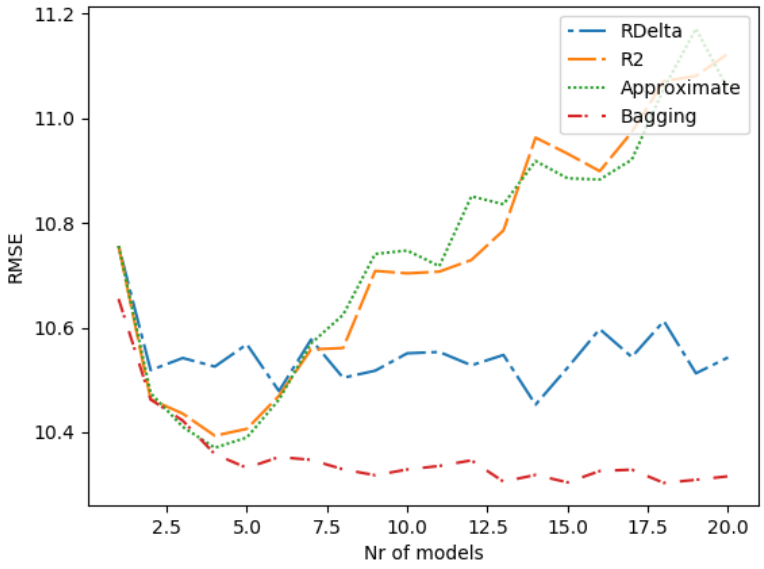
compared using a paired t-test to determine whether or not observed differences were statistically significant at a confidence level of 0.05. The results from the t-tests indicated significant differences between spatial and non-spatial results, as well as between different AdaBoost methods that train a different number of weak models. There were also statistical differences between all single neural network results and the bagging and AdaBoost results. When comparing the different AdaBoost models to bagging, bagging results were always statistically significantly different from all AdaBoost results.

6.4 Results and Discussion

Some interesting results regarding AdaBoost and bagging are presented in this section. For a comprehensive discussion of results for all methods and all fields, we refer to Chapter 7. Tables 6.2 and 6.3 show the results for bagging with four different subset ratios (0.2, 0.4, 0.6, and 0.8) using 20 weak models. These results are generally not statistically significantly different from each other, but there is a significant difference between the non-spatial and spatial results for the data sets with the same ratio. Because of this insignificant difference, a ratio of 0.2 was used



(a) Field sec35mid.



(b) Field sre1314.

Figure 6.2: RMSE values on two of the fields studied for bagging and the three different AdaBoost methods, with the number of models ranging from 1 to 20.

Table 6.3: Protein prediction RMSE results for bagging using four different subset selection ratios: 0.2, 0.4, 0.6, and 0.8. Each of the ratios was run with 20 weak models.

	sre1314		**sec35mid**		**davidson**		**carlin**	
	spatial	**non-spatial**	**spatial**	**non-spatial**	**spatial**	**non-spatial**	**spatial**	**non-spatial**
0.2	0.872	0.945	1.346	1.336	1.656	1.550	1.345	2.305
0.4	0.805	0.895	1.264	1.303	1.666	1.548	1.295	4.721
0.6	0.791	0.878	1.287	1.287	1.597	1.536	1.342	2.978
0.8	0.796	0.863	1.281	1.282	1.631	1.530	1.406	4.243

throughout the other experiments, as this is the least computationally expensive. Tables 6.4 and 6.5 show the average RMSE scores for yield and protein data using bagging and the three different AdaBoost implementations for 5, 10, and 20 models. Both the spatial and non-spatial data results are given. The best results are carried over into Table 6.6, which compares the simple FFNN results to the AdaBoost and bagging results.

In general, the original hypothesis that spatial results will improve performance, as stated in our preliminary paper [95], holds for the yield data. There is one notable exception for sec35mid. As shown in Table 6.4, the spatial AdaBoost model consistently performs worse than the AdaBoost model using the non-spatial data on this field. The spatial analysis only considers data points that belong to cells that have eight surrounding cells, as each data point has to have the same dimensions to be used by the predictive models. Because the field in question consists of two separated parts (Figure 3.2d), the number of data points reduces by 35%. There are two parts that lose the edge cells on the field [1], which explains the large reduction in points. This data point reduction could explain why the spatial analysis for this

[1] edge cells are the cells on the borders that are not fully surrounded by other cells

field is not as effective; however, this problem does not occur when applying bagging. This could indicate the weighting of the spatial data points negatively influences the outcome. There may be fewer outliers for the spatial prediction and by putting more emphasis on these fewer outliers, the overall predictive power goes down.

Similarly, for protein data, using spatial data does not always improve results. Table 3.1 shows that the number of data points for protein is much smaller than for yield. Certain spatial data sets could therefore suffer from the curse of dimensionality. The number of features becomes much larger when adding in the spatial information, but there may not be enough data to analyze the data properly, thereby negatively influencing predictive power. However, adding spatial data when using bagging does always result in a significant improvement (Table 6.3). This may indicate that adding in the spatial data helps the bagging models discover the underlying correlation between the data points.

A second important result to note is that applying AdaBoost and bagging improves over, or in two cases maintains, results from a single FFNN across all fields and data sets (Table 6.6), with one notable exception for bagging for non-spatial yield prediction of the field "carlinwest." We suspect this may be due to the nature of the field, it could be highly irregular in topographical features. Bagging seems to be unable to pick up on the spatial correlation of the data. This confirms our hypothesis that ensemble methods can improve predictive results. For yield, even if the prediction error only improves by one unit, this means that it is 60 pounds of yield per acre closer to the actual value. The results show that there is an improvement in error of 3 to 10 units, or 180 to 600 lbs of yield per acre, which is a non-negligible amount. The protein results are improved by smaller numbers, as this indicates the percentage of protein in a single grain, which is inherently a small number.

It also seems that combining a smaller number of weak models performs better

than increasing the number of models (Table 6.4, Table 6.5, Figure 6.2). This is in line with the idea that "many may be better than all" [137], where increasing the number of models may tend to increase the error, perhaps through some form of over-fitting. However, instead of training all the models and selecting from those models, simply training fewer models may have a similar effect while needing less time. Interestingly, the same does not always hold true for bagging. In many cases, the bagging ensemble improves as more models are added. Furthermore, bagging outperforms boosting for most data sets, contrary to what we expected. As bagging does not shift weight toward incorrectly classified data points, each model has the same level of influence over the final result. Keeping that in mind, it makes sense that adding more models would stabilize results, and consequently improve overall predictive quality.

When comparing the different AdaBoost methods, no one method seems to outperform the others. It is interesting to note that AdaBoost.R2 and AdaBoost.App both seem to work much better with a smaller number of models, whereas results for AdaBoost.RΔ vary across the number of models (Figure 6.2). This may be due to the way the model error ϵ_t is calculated and the individual points' weights are updated for AdaBoost.RΔ. As the probability distribution of incorrectly classified data points is updated based on their previous probability, the distribution may not be changed substantially. This could have resulted in relatively small differences between models' weights. Then, even if later models have a larger influence on the end result (i.e., a heavier weight), the difference in model weight may be small enough such that initial models, which should be correctly predicting easier data points, still play an important role in calculating the final values.

Finally, note the similarity in predictions between AdaBoost.R2 and AdaBoost.App, especially for yield predictions. We believe this may be due to improper tuning of the threshold value for AdaBoost.App, resulting in only a small number

of data points falling below that threshold, effectively resulting in very similar predictions for both models. When looking at Figure C.9, there is a clear difference in the performance of the two models. This indicates that the threshold was more appropriately set for the protein data for AdaBoost.App.

Table 6.4: Yield prediction RMSE results for the different AdaBoost methods and bagging, each training 5,10, and 20 weak models, across all fields using spatial and non-spatial data. Results in *italics* are carried over into table 6.6. **Bolded** results are the lowest scores for a field for boosting and bagging individually.

		AdaBoost.RΔ			**AdaBoost.R2**			**AdaBoost.App**			**Bagging**		
		5	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**
sre1314	**spatial**	9.674	9.546	***9.508***	9.828	11.095	11.923	9.920	11.018	11.938	9.432	9.295	***9.227***
	non-spatial	10.478	10.509	10.489	10.469	10.668	11.031	*10.462*	10.737	10.989	10.332	10.329	*10.316*
sec35mid	**spatial**	19.998	21.462	21.898	20.241	13.840	10.209	19.722	13.898	*10.092*	5.355	5.049	***4.856***
	non-spatial	8.998	8.657	8.612	8.591	8.545	9.077	8.471	***8.388***	9.026	7.529	7.388	*7.297*
davidson	**spatial**	8.008	7.929	7.889	***7.815***	8.140	8.480	7.821	7.990	8.467	8.237	7.747	***7.547***
	non-spatial	14.049	13.766	13.778	13.433	13.865	14.573	*13.420*	13.906	14.590	14.073	*13.896*	13.937
carlin	**spatial**	***5.425***	5.467	5.652	5.638	6.779	7.885	5.691	6.795	8.436	6.751	6.384	***5.770***
	non-spatial	7.893	6.850	6.877	*6.437*	6.596	7.570	6.625	6.781	7.576	180.166	181.009	*175.947*

Table 6.5: Protein prediction RMSE results for the different AdaBoost methods and bagging, each training 5,10, and 20 weak models, across all fields using spatial and non-spatial data. Results in *italics* are carried over into table 6.6. **Bolded** results are the lowest scores for a field for boosting and bagging individually.

		AdaBoost.RΔ			**AdaBoost.R2**			**AdaBoost.App**			**Bagging**		
		5	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**
sre1314	**spatial**	0.981	0.977	0.994	0.956	***0.934***	0.974	0.970	0.954	0.974	0.993	0.895	***0.872***
	non-spatial	1.019	0.994	0.981	0.978	*0.971*	0.985	0.983	0.992	1.132	1.012	*0.994*	0.995
sec35mid	**spatial**	2.185	1.920	*1.850*	2.211	2.199	2.210	2.171	2.119	2.356	1.531	1.388	*1.346*
	non-spatial	1.636	1.602	***1.320***	1.609	1.494	1.377	1.498	1.412	1.406	1.482	***1.334***	1.336
davidson	**spatial**	1.706	1.720	1.849	1.709	1.737	1.721	1.746	*1.701*	1.680	2.004	1.743	*1.656*
	non-spatial	1.551	***1.527***	1.926	1.594	1.627	1.754	1.609	1.686	1.688	1.630	1.564	***1.550***
carlin	**spatial**	1.739	1.676	1.891	1.852	1.599	1.454	1.730	1.527	*1.426*	1.406	1.376	***1.349***
	non-spatial	2.741	2.657	3.485	2.696	5.545	6.516	***1.244***	6.236	4.641	*1.661*	2.481	2.305

Table 6.6: Yield and protein prediction RMSE results for a single FFNN, compared to the best AdaBoost result from tables 6.4 and 6.5.

		sre1314		**sec35mid**		**davidson**		**carlin**	
		Yield	**Protein**	**Yield**	**Protein**	**Yield**	**Protein**	**Yield**	**Protein**
Spatial	**FFNN**	12.546	1.184	14.606	1.964	17.735	1.641	9.712	1.307
	AdaBoost	9.508	0.934	10.092	1.850	7.815	1.680	5.425	1.454
	Bagging	9.227	0.872	4.856	1.346	7.547	1.656	5.770	1.349
Non-spatial	**FFNN**	12.141	1.187	14.916	2.320	22.836	6.945	10.607	7.452
	AdaBoost	10.489	0.971	8.388	1.320	13.766	1.527	6.596	1.244
	Bagging	10.316	0.994	7.297	1.334	13.896	1.550	175.947	1.661

CHAPTER SEVEN

COMPARITIVE ANALYSIS ON REAL FARMS

This chapter explores the results for all six models for the different data sets. An overview of the RMSE results for yield and protein is presented in Tables 7.1 and 7.2 respectively. These tables show the RMSE values for the FFNN (results taken from Chapter 6), and the SAE, as well as results for the four different ensemble methods. The results shown for the ensemble methods are the best results out of the 20 runs with different numbers of weak models. Appendix B shows all of the plots generated for the four different ensemble methods across the different number of models. The RMSE values are plotted for 1 to 20 models, showing how the ensemble methods perform as the number of models increases. The original linear and non-linear models are not shown, as Chapter 5 showed that the spatial machine learning approaches significantly improve results overall.

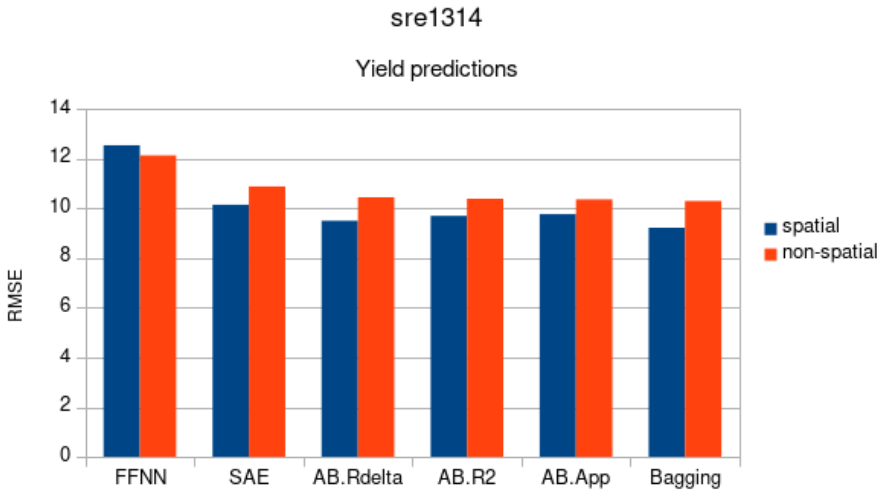
Due to the large amount of results presented, we will first look at all the fields separately and from these individual analyses draw an overall conclusion. For the purpose of readability, bar charts were created for each field showing the RMSE results for yield and protein in separate graphs across the different models. For the ensemble methods, the best results across the 20 models are used, as shown in Tables 7.1 and 7.2. These tables also show the amount of models found to have the best performance for each of the ensembles.

Table 7.1: Yield prediction RMSE results for all implemented models, for both spatial and non-spatial data. The number of models that provided the optimal results for the four ensemble methods is given in column "#mod". The best predictive results for a field are indicated in **bold**, and the best results for spatial and non-spatial separately are shown in *italics*.

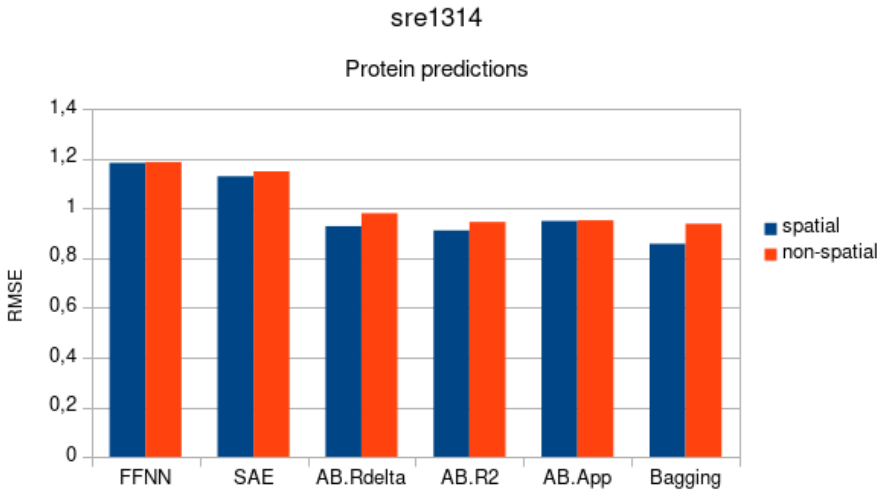
	sre1314				**sec35mid**				**davidson**				**carlin**			
	spatial		**non-spatial**		**spatial**		**non-spatial**		**spatial**		**non-spatial**		**spatial**		**non-spatial**	
	#mod	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**
FFNN	n/a	12.546	n/a	12.141	n/a	14.606	n/a	14.916	n/a	17.735	n/a	22.836	n/a	9.712	n/a	10.607
SAE	n/a	10.15	n/a	10.89	n/a	8.95	n/a	10.09	n/a	12.06	n/a	16.03	n/a	9.15	n/a	9.93
AB.RΔ	20	9.508	14	10.453	12	16.980	19	8.527	16	7.738	8	13.716	5	5.425	11	6.858
AB.R2	3	9.704	4	10.393	19	10.154	6	8.381	5	7.815	4	13.320	3	**5.419**	5	*6.437*
AB.App	2	9.775	4	10.370	19	10.028	8	8.256	4	7.819	4	*13.290*	2	5.475	6	6.452
Bagging	20	***9.227***	18	*10.303*	20	**4.856**	19	*7.287*	20	***7.547***	11	13.841	19	5.717	4	122.150

Table 7.2: Protein prediction RMSE results for all implemented models, for both spatial and non-spatial data. The number of models that provided the optimal results for the four ensemble methods is given in column "#mod". The best predictive results for a field are indicated in **bold**, and the best results for spatial and non-spatial separately are shown in *italics*.

	sre1314				**sec35mid**				**davidson**				**carlin**			
	spatial		**non-spatial**		**spatial**		**non-spatial**		**spatial**		**non-spatial**		**spatial**		**non-spatial**	
	#mod	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**	**#mod**	**RMSE**
FFNN	n/a	1.184	n/a	1.187	n/a	1.964	n/a	2.320	n/a	1.641	n/a	6.945	n/a	1.307	n/a	7.452
SAE	n/a	1.13	n/a	1.15	n/a	1.96	n/a	1.97	n/a	*1.61*	n/a	1.67	n/a	*1.29*	n/a	1.37
AB.RΔ	14	0.929	20	0.981	16	1.829	15	1.589	9	1.706	10	***1.527***	10	1.676	3	1.313
AB.R2	8	0.912	12	0.946	3	2.103	19	1.382	18	1.682	2	1.574	18	1.450	1	1.337
AB.App	7	0.950	7	0.953	8	2.116	17	1.370	16	1.672	2	1.575	19	1.392	5	***1.244***
Bagging	19	***0.858***	17	*0.939*	9	*1.340*	19	***1.298***	15	1.649	16	1.540	15	1.319	12	1.550



(a) Yield results.



(b) Protein results.

Figure 7.1: Yield and protein for field "sre1314".

7.1 Field sre1314

As can be seen in Figure 7.1, spatial bagging performs better for both yield and protein; however, none of the ensemble results vary much from one another. Even though the results are statistically significantly different, they are not much better

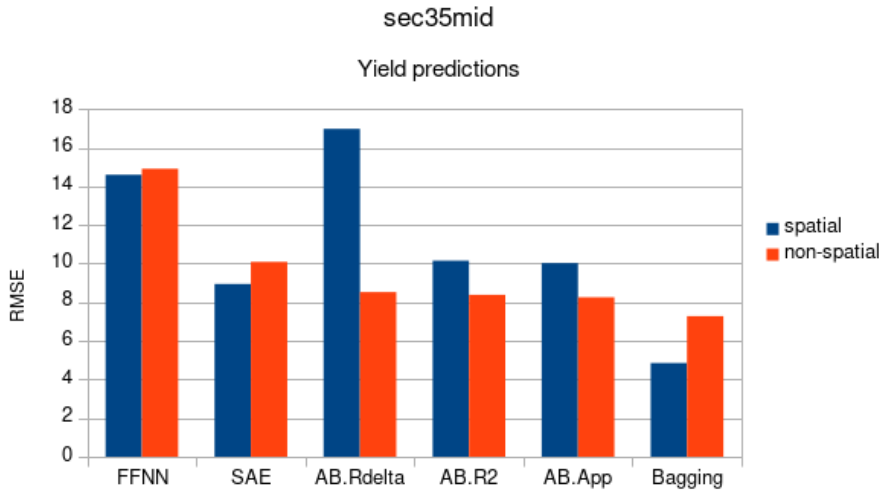
or worse than each other. The ensemble methods and SAE do perform better than the FFNN, confirming our hypotheses concerning single, shallow neural networks and deep networks as well as ensembles.

Overall, adding in spatial data improves the results for both yield and protein, although not by much. This is especially interesting as the original data set contains the most data points out of all four fields. However, adding in the spatial information decreases the data set size by approximately 35%, while increasing the number of features by almost 8 times the original amount (for the 8 surrounding cells, see Figure 3.3). This could be creating an imbalance in the data set, thus hindering predictive performance.

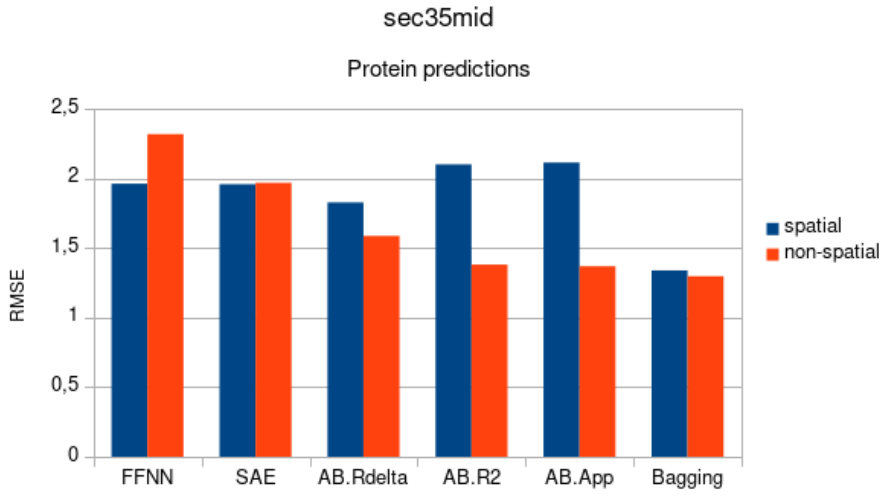
When looking at how the ensemble methods behave as the number of weak models increases, bagging generally improves as the number of weak models increases. All four ensemble methods improve in performance as the number of models increases from 1. However, AdaBoost.R2 and Approximate AdaBoost show a decrease in performance as the number of models starts to increase. This is especially apparent for the spatial and non-spatial yield data. It may be that the neural networks are overfitting the data on the training set, thus performing very poorly on the test set.

7.2 Field sec35mid

As mentioned in Chapter 6, the spatial data decreases predictive power for the three AdaBoost implementations. We speculated this may be due to the shape of the field and the larger loss of data, a 35% decrease in the number of yield points, as well as due to the nature of AdaBoost. The latter is deemed to be the case due to the increase in performance of bagging when using spatial data. This conjecture is further confirmed when looking at the results for the FFNN and SAE, as both models are improved by implementing spatial data.



(a) Yield results.



(b) Protein results.

Figure 7.2: Yield and protein for field "sec35mid".

Overall, the SAE and ensemble methods once again improve the results for yield prediction. However, for this field, there is a noticeable difference in performance between bagging and the other methods. [1] As a matter of fact, bagging performs statistically significantly better for this specific field for both yield and protein. This

[1]This becomes especially clear when looking at Figures C.3 and C.4.

figure also shows interesting trends for the different AdaBoost implementations as the number of models changes. There is no consistent trend to be discerned between the four different data sets.

7.3 Field davidsonmidwest

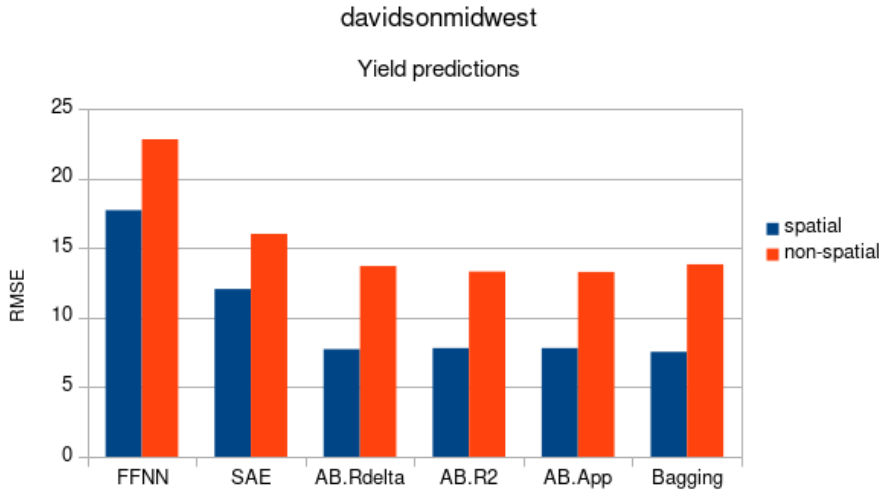
Figure 7.3a shows an overall improvement of results when using spatial yield data, whereas the same is not true for protein results (Figure 7.3b). The FFNN and SAE both perform better with spatial data, especially the FFNN that has a strong decrease in RMSE. However, as the FFNN results are unstable, even with 10-fold cross validation, the relatively large RMSE score may just be due to this specific iteration of the network.

This field is especially interesting when looking at the different ensemble results (Figures C.5 and C.6). Here, bagging performs very similarly to the AdaBoost methods, and AdaBoost outperforms bagging in some cases. It is important to note that the differences in RMSE values between the different ensembles are very small, less than a single unit of yield and protein, so bagging is still performing well. The main difference is that the AdaBoost implementations are performing equally well on this specific field.

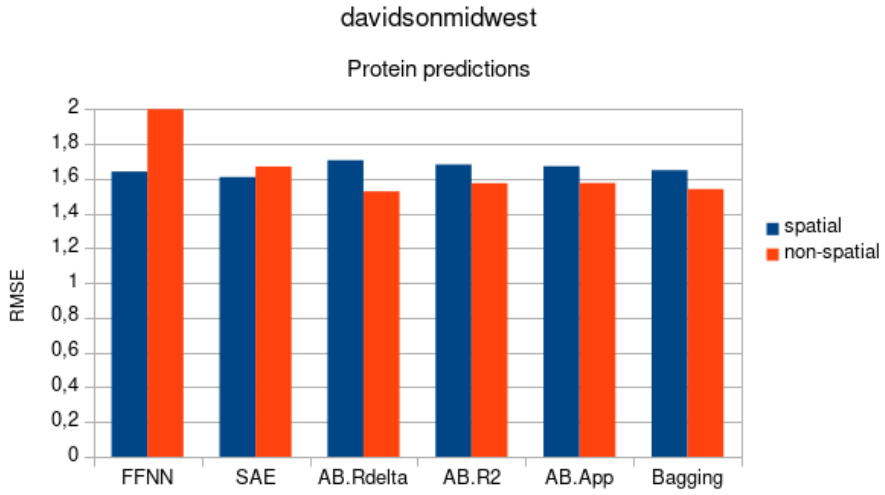
7.4 Field carlinwest

Once again, adding in spatial data improves yield predictions on field "carlinwest". And similar to field "davidsonmidwest", protein prediction has varying results, with the FFNN being the only model that is vastly improved by adding in the spatial analysis.

However, this field is the only exception to the overall well-performing bagging



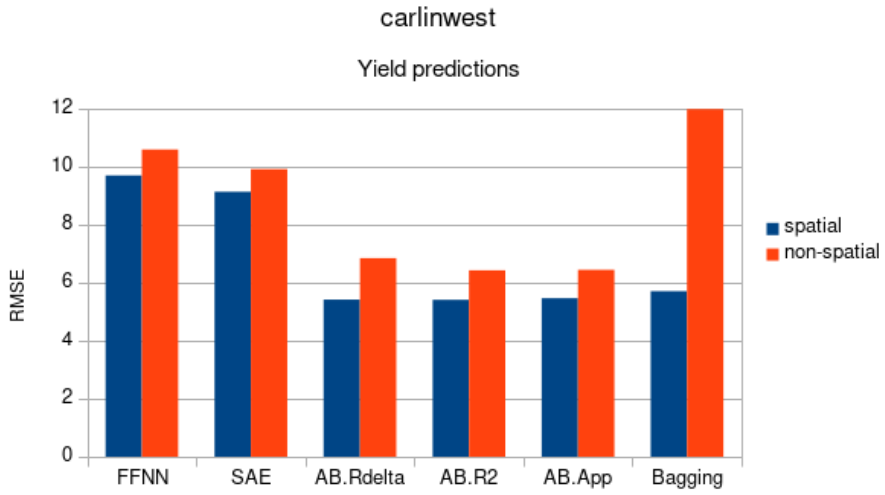
(a) Yield results.



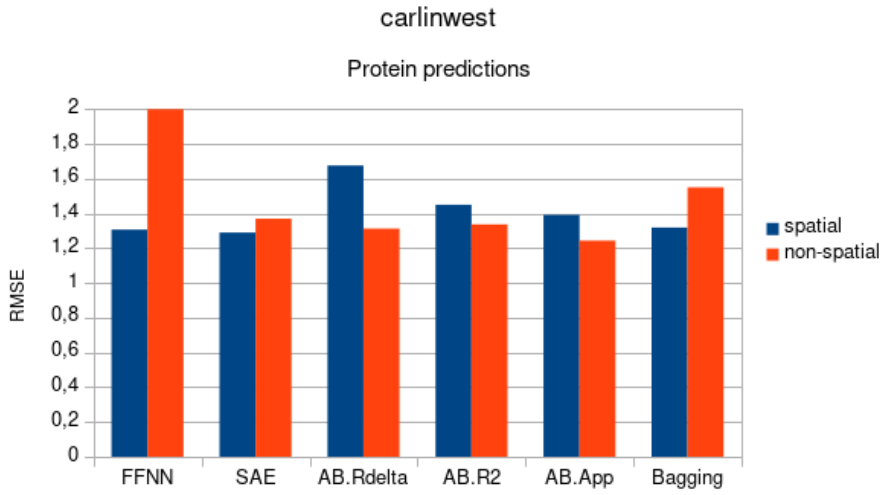
(b) Protein results.

Figure 7.3: Yield and protein for field “davidsonmidwest”.

ensemble. The RMSE value for non-spatial yield predictions for bagging is more than 10 times the number of the next highest RMSE value. This is an interesting anomaly, as this is not the largest or smallest data set, but lies in the middle, and the field itself is a regular rectangular shape. The bagging performance for the spatial yield data set also has a large peak when aggregating two and three weak models; however,



(a) Yield results.



(b) Protein results.

Figure 7.4: Yield and protein for field “carlinwest”.

the results greatly improve after that. Seeing this indicates the non-spatial bagging method may not be picking up on the spatial correlations within the field.

7.5 Overview of All Four Field

Throughout most data sets, our hypothesis that adding in spatial information to the yield data would improve results holds. For three out of four fields, yield prediction is improved across the board. The fourth field, "sec35mid", has poor performance with the combination of spatial data and the three AdaBoost methods. As mentioned in Chapter 6, this could be due to the structure of the field and the weighted nature of AdaBoost. When looking at the protein graphs, the results are less consistent. This is in line with what we expected. Due to the size of the protein data sets, adding in many extra features creates an imbalanced data set, making the learner incapable of fitting the data correctly. This theory is corroborated by the fact that the only protein predictions that are not impacted negatively for any of the spatial models are for the largest protein data set (Table 3.1).

Generally, the different AdaBoost methods perform similarly. Especially AdaBoost.R2 and Approximate AdaBoost tend to follow similar trends. This may mean that "classifying" a point as correct within a certain boundary, as opposed to simply taking the difference between predicted and actual values into account for all points, does not have a large influence on how the algorithm performs. However, this may also be because the threshold value was not tuned appropriately for each data set. When looking at how these two methods perform as the number of models increases, for over half the data sets, the predictive quality seems to go down as the number of models goes up. There is usually a drop in RMSE values when more than a single model is used. The only exception is for the "carlinwest" protein data set. However, the "davidsonmidwest" protein data has a similar result, there is only a small dip in RMSE prediction when using two weak models as opposed to one, but the models' predictive power decreases after that. These are the two smallest data

sets, which could explain why the ensemble methods are not working as well, there might simply not be enough data.

AdaBoost.RΔ also uses a threshold value, but then classifies the points as true or false, not taking the difference in values of the "falsely" labeled data points into account. It performs inconsistently across data sets, and its results vary the most as the number of models changes. When looking at the latter results, the RΔ approach does not seem to follow a trend line of any kind. This is in contrast to bagging, which generally has a downward trend in RMSE as the number of weak learners goes up. Overall, AdaBoost.RΔ seems to be the weakest of the ensemble methods when considering both RMSE results across data sets and general consistency.

The stacked auto-encoder does outperform the ensemble methods on one of the protein data sets but is not as consistent. This confirms our hypothesis that ensemble methods would be better suited to the problem at hand than a deep network. However, bagging outperforms boosting on all but one data set (non-spatial yield predictions of "carlinwest"), unlike what we expected. When the surrounding data is added to this particular data set, bagging does find an underlying structure and performs similarly to the other models. Furthermore, bagging improves significantly when adding spatial data for all 8 data sets (Tables 6.2 and 6.3). Overall, the spatial bagging approach is the most consistent of all models, confirming the hypothesis that there would be a single model that has good predictive performance across all data sets.

CHAPTER EIGHT

CONCLUSION

8.1 Discussion

Precision agriculture has the ability to improve food production. In the overarching research of this thesis, a workflow is created to optimize farmers' net return by creating optimal nitrogen prescriptions based on expected yield and protein (Chapter 3). This not only has the potential to improve yield and net return, but can also decrease the amount of nitrogen applied to fields, thus positively impacting the environment. There are two key elements that make the decision-making process unique: the use of low-cost data and the spatio-temporal updating aspect. This thesis specifically looked at two aspects of the workflow: experimental prescription map design and yield and protein prediction.

Chapter 4 addresses the experimental prescription map design. Variable rate application is an important aspect of PA, trying to reduce the amount of fertilizer needed across a field while simultaneously increasing yield, protein production, and net return. To be able to determine the amount of nitrogen to be applied adequately, field-specific experiments have to be performed. These experiments involve the creation of a nitrogen prescription map that overlays a grid on the field at hand and looks at previous year yield and protein values within each cell to determine what bin the cell belongs to. Then, nitrogen rates are randomly assigned to cells with the goal of having an even distribution of nitrogen values across each bin, thus providing a good experimental basis to examine the influence of nitrogen rates on different parts of the field. However, these prescription maps often contain large jumps between consecutive cells, putting strain on the farming equipment.

In our experiments, a Genetic Algorithm was applied successfully to minimize the jumps between cells, while maintaining stratification, thus supporting our hypothesis. With the use of an appropriate multi-objective fitness function, the GA was able to reduce jumps between cells, with only a slight loss of stratification on occasion. This was accomplished by setting the jump score weight higher than the stratification score weight and by implementing swap mutation. Swap mutation seemed to speed up convergence when compared to scramble mutation, which needed more generations and seems to converge to a higher fitness score. The resulting reduction in the stratification score might be avoided by giving equal weight to the two objectives in the fitness function. However, doing this decreased convergence rate, and it took much longer to reduce the jump score, which is undesirable when being used in practice. Scramble mutation performed better than swap mutation in this scenario. Overall, the GA performs well for this particular application and achieved the set goal.

PA has seen an increase in computationally-based research, especially involving machine learning. By applying machine learning to train predictive models, fertilization prescription maps and yield/protein prediction could become increasingly accurate, resulting in higher productivity. Chapters 5 and 6 respectively look at deep learning and ensemble methods from machine learning to address yield and protein prediction. Despite the amount of previous work using machine learning, there has been no significant research into applying deep learning techniques or ensemble methods to PA.

The research presented in Chapter 5 compared multiple regression, shallow feed-forward networks and stacked autoencoders (a deep method), the latter two both nonspatially and spatially, to generate models for yield and protein prediction. Our results confirm our hypothesis that spatial stacked autoencoders and feed-forward

networks seem to provide significantly more accurate results than the competing methods on the fields studied. It is interesting to note that there were not always significant improvements in results between the shallow and deep networks, contrary to what we expected.

In Chapter 6, a simple single layer FFNN is compared to three different AdaBoost methods and bagging, each using single layer FFNNs as the weak learner, and using both spatial and non-spatial data to train the models. Our results once again confirm that spatial data tends to improve predictions, but the amount of available data can negatively influence performance. Consistent with findings from other studies as well as in line with out hypothesis, ensemble methods improved predictions compared to a single model. In our case, yield predictions improved by between 3 to 10 units, meaning the error was reduced by 180 to 600 lbs of yield per acre. When comparing results from different AdaBoost runs, the three implementations performed similarly. We expected boosting to be the best performing ensemble method; however, bagging outperformed AdaBoost for most fields. Especially spatial bagging had consistent performance across all data sets.

8.2 Future Work

The work shown in this thesis is only a small portion of possible directions for maximizing net return and minimizing environmental impact. Other PA approaches that are feasible at a relatively low cost include weed detection and crop disease identification. For such approaches it could be useful to collaborate with an optics lab to use hyperspectral imaging technologies. Aside from these more general directions, there are several potential future work opportunities for the prescription map and yield and protein prediction aspects of this thesis.

First, in terms of the genetic algorithm, we would like to compare different GA

operators in more detail, as well as explore how changing the weight of the multi-objective fitness function further influences results. We will continue to explore the impact of additional constraints in generating prescription maps, such as navigational constraints for the spreader/sprayer. We will also consider application of herbicides. The next step is to deploy this process for actual farmer use. This then becomes part of the workflow (Figure 3.1) and makes it possible to gather more data, while further decreasing cost to the farmers by lessening strain on their equipment. Eventually, the process will also be adjusted to be used when generating the optimized fertilization prescription maps. We did not address any research related to the creation of these optimal prescriptions, but it is an important overarching goal of the project. This part of the process will need different constraints, and stratification will no longer be part of the fitness function.

When it comes to the yield and protein data analysis, there are multiple interesting aspects that came to light throughout this thesis. First, there are several methods of sampling spatial context not described in this work that are worth exploring, such as Equal Distance and Random Selection. We would like to evaluate the relative performance of including these different types of spatial context. Furthermore, the lack of data points for some of the spatial analysis, could be addressed by performing other forms of spatial analysis such as kriging [21].

When looking at how the deep learning method held up against a shallow neural network, the similar performance of the ANN and SAE warrant more investigation into the workings of both models. As the currently implemented stacked autoencoder was originally designed for different research, it may be useful to see whether the autoencoder can be modified to better suit this specific problem. Another interesting aspect to consider is that the structures of the ANN and SAE are adapted depending on the data being analyzed. In other words, unique models are defined for each unique

field. An area of research receiving attention is "transfer learning" where knowledge learned in one domain (i.e., a particular field) is reused in the context of a different problem (i.e., another field). It may be interesting to explore how this could decrease training complexity or even improve overall accuracy.

For future research directions as far as ensembles are concerned, we would like to look at random forests (RF), as these are widely applied for yield prediction. Implementing an RF and adapting the random forest structure to neural networks could improve results further. In terms of the novel AdaBoost.App method, the results indicated the importance of correctly setting the threshold value when calculating the error. To this end, we would like to research ways to automatically set the threshold value. Aside from implementing the method proposed by [135], devising a method using the values of bias nodes in a neural network might be a good way to determine the magnitude of the threshold value. Creating an automatic threshold calculation avoids the addition of an extra tuning parameter and could improve overall predictive power of the model. In order to properly evaluate the performance of the model compared to other AdaBoost implementations, a study using benchmark data as well as the PA data would have to be performed.

In the grand scheme of the project, the next step for these yield and protein prediction methods is to incorporate the results into the economic model for actual profit optimization (Figure 3.1). This thesis does not address how yield and protein prediction relates back to fertilizer application. As mentioned previously, predicting the optimal fertilization rate is another important aspect of the overall project. We are investigating the extent to which we can use these models to facilitate creating optimized fertilization prescriptions.

8.3 Contributions

In this thesis two main contributions were made. The first solves the problem of large rate jumps between application cells, putting strain on farmers' equipment. The second contribution improves yield and protein prediction for an improved net return calculation. Aside from these two key contributions, several smaller questions were answered.

As the first main contribution, this work presented a novel application of a GA to a relevant problem in PA. The central research question asks whether a GA can be used successfully to reduce the number and magnitude of rate jumps between consecutive cells of a prescription map, while still maintaining stratification across the yield and protein bins? The short answer is "yes," confirming our hypothesis. Furthermore, two different mutation operators were implemented and a change in weight between the two objectives of the fitness functions was explored. When emphasizing the minimization of the jump score, swap mutation had the best performance. This implementation had the best results overall, with faster convergence and a larger decrease in jump score. When giving both jump and stratification equal weight, scramble mutation had slightly better performance, but convergence was slow and not reached within a 1000 generations.

The determination of a single predictive model that performs well for yield and protein prediction is the second key contribution of this thesis. An in-depth analysis was provided of several regression techniques applied to the problem of yield and protein prediction, as well as a comparison of using spatial and non-spatial data. Six machine learning methods were investigated: a feed-forward neural network, a stacked autoencoder, three different AdaBoost implementations with neural networks, and a bagging ensemble of neural networks. One of the proposed AdaBoost methods was a

novel version looking to combine ideas from two other methods. The results showed that spatial bagging performed well for both yield and protein prediction across all four fields. It did not always have the "best" performance, but there was no significant difference in results when this was the case.

In terms of predictive analysis, a few other conclusions can be drawn. Spatial analysis generally improved results across the models. The main exception concerns protein data, which can be explained by the small size of the data sets. Multiplying the number of features eight times while reducing the already small number of data points, creates a sparse data set, potentially hurting a model's ability to find the underling function. Lastly, ensemble methods seem to have a more consistent performance than deep learning for this specific problem. This could be due to the reduction in variance using ensemble methods can provide, or it could simply mean that an SAE has difficulty detecting the underlying correlations in the data; however, this seems unlikely.

REFERENCES CITED

[1] Adapt-n. http://www.adapt-n.com/. Accessed: 2019-02-01.

[2] Efc systems. http://www.efcsystems.com/. Accessed: 2019-02-01.

[3] EFC Systems agronomic planning and sustainability. http://www.efcsystems.com/index.php/agronomicplanningandsustainability/. Accessed: 2019-02-01.

[4] EFC Systems precision agronomy. http://www.efcsystems.com/index.php/fieldalytics/. Accessed: 2019-02-01.

[5] Trimble Ag farmer solutions. https://agriculture.trimble.com/software/farmers/. Accessed: 2019-02-01.

[6] R. Alvarez. Predicting average regional yield and production of wheat in the argentine pampas by an artificial neural network approach. *European Journal of Agronomy*, 30(2):70–77, 2009.

[7] SAS Software Analytics. *SAS/STAT(R) 13.2 User's Guide*. Accessed: 2019-03-01.

[8] Thomas M Banhazi, H Lehr, JL Black, H Crabtree, P Schofield, M Tscharke, and D Berckmans. Precision livestock farming: an international review of scientific and commercial aspects. *International Journal of Agricultural and Biological Engineering*, 5(3):1–9, 2012.

[9] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 153–160, Cambridge, MA, USA, 2006. MIT Press.

[10] Alberto Bertoni, Paola Campadelli, and M Parodi. A boosting algorithm for regression. In *International Conference on Artificial Neural Networks*, pages 343–348. Springer, 1997.

[11] J Martin Bland and Douglas G Altman. Multiple significance tests: the bonferroni method. *British Medical Journal*, 310(6973):170, 1995.

[12] Rodolfo Bongiovanni and James Lowenberg-DeBoer. Precision agriculture and sustainability. *Precision Agriculture*, 5:359–387, August 2004.

[13] BAM Bouman. Crop modelling and remote sensing for yield prediction. *NJAS–Wageningen Journal of Life Sciences*, 43(2):143–161, 1995.

[14] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[15] Leo Breiman et al. Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383, 1996.

[16] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Blaisdell, 1969.

[17] Giorgia Bucci, Deborah Bentivoglio, and Adele Finco. Precision agriculture as a driver for sustainable farming systems: State of art in literature and research. *Quality–Access to Success*, 19:114–121, 2018.

[18] Hong Cheng, Lutz Damerow, Yurui Sun, and Michael Blanke. Early yield prediction using image analysis of apple fruit and tree canopy features with neural networks. *Journal of Imaging*, 3(1):6–29, 2017.

[19] Anna Chlingaryan, Salah Sukkarieh, and Brett Whelan. Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review. *Computers and Electronics in Agriculture*, 151:61–69, 2018.

[20] Roger Cousens. A simple model relating yield loss to weed density. *Annals of Applied Biology*, 107(2):239–252, 1985.

[21] Noel Cressie. The origins of kriging. *Mathematical Geology*, 22(3):239–252, 1990.

[22] Zhenling Cui, Fusuo Zhang, Xinping Chen, Zhengxia Dou, and Junliang Li. In-season nitrogen management strategy for winter wheat: Maximizing yields, minimizing environmental impact in an over-fertilization context. *Field Crops Research*, 116(1):140–146, 2010.

[23] Pdraig Cunningham, John Carney, and Saji Jacob. Stability problems with artificial neural networks and the ensemble solution. *Artificial Intelligence in Medicine*, 20(3):217–225, 2000.

[24] Snehal Dahikar, Sandeep Rode, and Pramod Deshmukh. An artificial neural network approach for agricultural crop yield prediction based on various parameters. *International Journal of Advanced Research in Electronics and Communication Engineering*, 4(1):94–98, 2015.

[25] Yash Dang, Roger Lawes, Nigel Metz, and Brett Whelan. *Make variable-rate application pay*. Grains Research and Development Corporation, March 2012.

[26] EA Davidson, Mark B David, James N Galloway, Christine L Goodale, Richard Haeuber, John A Harrison, Robert W Howarth, Dan B Jaynes, R Richard Lowrance, Nolan B Thomas, et al. Excess nitrogen in the us environment: trends, risks, and solutions. *Issues in Ecology*, (15), 2011.

[27] Eric M. Delmelle. Spatial sampling. In M.M. Fischer and P. Nijkamp, editors, *Handbook of Regional Science*, chapter 70, pages 1385–1399. Springer-Verlag Berlin Heidelberg, 2014.

[28] HJ Dennis and Wilhelm T Nell. *Precision irrigation in South Africa*. Technical Centre for Agricultural and Rural Cooperation, 2006.

[29] Thomas G Dietterich. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer, 2000.

[30] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1470–1477. IEEE, 1999.

[31] Charles T Driscoll, David Whitall, John Aber, Elizabeth Boyer, Mark Castro, Christopher Cronan, Christine L Goodale, Peter Groffman, Charles Hopkinson, Kathleen Lambert, et al. Nitrogen pollution in the northeastern united states: sources, effects, and management options. *BioScience*, 53(4):357–374, 2003.

[32] Harris Drucker. Improving regressors using boosting techniques. In *International Conference on Machine Learning*, volume 97, pages 107–115, 1997.

[33] Scott Drummond, Anupam Joshi, and Kenneth Sudduth. Application of neural networks: Precision farming. In *IEEE World Congress on Computational Intelligence*, pages 211–215, 1998.

[34] Russell Eberhart and James Kennedy. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[35] Bradley Efron and Robert Tibshirani. *An Introduction To The Bootstrap*, volume 89, page 436. 1993.

[36] A. E. Eiben and J. E. Smith. *Fitness, Selection, and Population Management*, pages 79–98. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[37] A. E. Eiben and J. E. Smith. *Representation, Mutation, and Recombination*, pages 49–78. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[38] Konstantinos P Ferentinos and Theodore A Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Computer Networks*, 51(4):1031–1051, 2007.

[39] Jeferson Lobato Fernandes, Nelson Francisco Favilla Ebecken, and Jlio Csar Dalla Mora Esquerdo. Sugarcane yield prediction in brazil using ndvi time series and neural networks ensemble. *International Journal of Remote Sensing*, 38(16):4631–4644, 2017.

[40] Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, March 1995.

[41] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[42] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, volume 96, pages 148–156, 1996.

[43] Deepak Ganesan, Sylvia Ratnasamy, Hanbiao Wang, and Deborah Estrin. Coping with irregular spatio-temporal sampling in sensor networks. *Computer Communications Review*, 34:125–130, 2004.

[44] Robin Gebbers and Viacheslav I. Adamchuk. Precision agriculture and food security. *Science*, 327(5967):828–831, 2010.

[45] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Autoencoders*, chapter 5, pages 499–523. MIT Press, 2017.

[46] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2017.

[47] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Machine Learning Basics*, chapter 5, pages 97–105. MIT Press, 2017.

[48] C. Groos, N. Robert, E. Bervas, and G. Charmet. Genetic analysis of grain protein-content, grain yield and thousand-kernel weight in bread wheat. *Theoretical and Applied Genetics*, 106(6):1032–1040, Apr 2003.

[49] Kyoungnam Ha, Sungzoon Cho, and Douglas MacLachlan. Response models based on bagging neural networks. *Journal of Interactive Marketing*, 19(1):17–30, 2005.

[50] Randy L. Haupt and Sue Ellen Haupt. *Practical genetic algorithms*. Taylor & Francis, 2004.

[51] Guang-Hua He, Lei Hou, De-Mou Li, Xiao-Ying Luo, Guo-Qing Niu, Mei Tang, and Yan Pei. Prediction of yield and yield components in hybrid rice by using molecular markers. *Acta Genetica Sinica*, 29(5):438–444, May 2002.

[52] D. O. Hebb. The organization of behavior: A neuropsychological theory. *Science Education*, 34(5):336–337, 1950.

[53] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *IEEE International Joint Conference on Neural Networks*, pages 65–93, 1989.

[54] Stien Heremans, Qinghan Dong, Beier Zhang, Lieven Bydekerke, and Jos Van Orshoven. Potential of ensemble tree methods for early-season prediction of winter wheat yield from short time series of remotely sensed normalized difference vegetation index and in situ meteorological data. *Journal of Applied Remote Sensing*, 9:9–20, 2015.

[55] Charles R. Hicks and Kenneth V. Turner. *Fundamental Concepts in the Design of Experiments*. Oxford University Press, 1999.

[56] John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[57] Hao Hu and Shaowen Wang. Estimating large-scale crop yields with recurrent neural networks. In *American Association of Geographers: Symposium on CyberGIS and Spatial Data Science*, 2018.

[58] Zehui Jiang, Chao Liu, Nathan P Hendricks, Baskar Ganapathysubramanian, Dermot J Hayes, and Soumik Sarkar. Predicting county level corn yields using deep long short term memory models. *ArXiv*, 2018.

[59] Kayuki Kaizzi, John Byalebeka, Onesmus Semalulu, Isaac Newton Alou, Williams Zimwanguyizza, Angella Nansamba, Emmanuel Odama, Patrick Musinguzi, Peter Ebanyat, Theodora Hyuha, Appollo K. Kasharu, and Charles Wortmann. Optimizing smallholder returns to fertilizer use: Bean, soybean and groundnut. *Field Crops Research*, 127:109–119, 2012.

[60] Andreas Kamilaris and Francesc X. Prenafeta-Bold. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90, 2018.

[61] Monisha Kaul, Robert L Hill, and Charles Walthall. Artificial neural networks for corn and soybean yield prediction. *Agricultural Systems*, 85(1):1–18, 2005.

[62] Sami Khanal, John Fulton, Andrew Klopfenstein, Nathan Douridas, and Scott Shearer. Integration of high resolution remotely sensed data and machine learning techniques for spatial prediction of soil properties and corn yield. *Computers and Electronics in Agriculture*, 153:213–225, 2018.

[63] Myoung-Jong Kim and Dae-Ki Kang. Ensemble with neural networks for bankruptcy prediction. *Expert Systems with Applications*, 37(4):3373–3379, 2010.

[64] Brad Koch, R Khosla, WM Frasier, DG Westfall, and D Inman. Economic feasibility of variable-rate nitrogen application utilizing site-specific management zones. *Agronomy Journal*, 96(6):1572–1580, 2004.

[65] Alexandra N Kravchenko and Donald G Bullock. Correlation of corn and soybean grain yield with topography and soil properties. *Agronomy Journal*, 92(1):75–83, 2000.

[66] K. Kuwata and R. Shibasaki. Estimating crop yields with deep learning and remotely sensed data. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 858–861, July 2015.

[67] WE Larson, JA Lamb, BR Khakural, RB Ferguson, and GW Rehm. Potential of site-specific management for nonpoint environmental protection. In *The State of Site-Specific Management for Agriculture*, pages 337–367. American Society of Agronomy, Crop Science Society of America, Soil Science Society of America, 1997.

[68] Patrick G Lawrence, Lisa J Rew, and Bruce D Maxwell. A probabilistic bayesian framework for progressively updating site-specific recommendations. *Precision Agriculture*, 16(3):275–296, 2015.

[69] Steve Lawrence, C Lee Giles, and Ah Chung Tsoi. Lessons in neural network training: Overfitting may be harder than expected. In *Innovative Applications of Artificial Intelligence Conference*, pages 540–545. Association for the Advancement of Artificial Intelligence, 1997.

[70] Niklaus Lehmann and Robert Finger. Optimizing whole-farm management considering price and climate risks. In *123rd European Association of Agricultural Economists Seminar*, February 2012.

[71] Ainong Li, Shunlin Liang, Angsheng Wang, and Jun Qin. Estimating crop yield from multi-temporal satellite data using multivariate regression and neural network techniques. *Photogrammetric Engineering & Remote Sensing*, 73(10):1149–1157, 2007.

[72] Jessica Lindblom, Christina Lundström, Magnus Ljung, and Anders Jonsson. Promoting sustainable intensification in precision agriculture: review of decision support systems development and strategies. *Precision Agriculture*, 18(3):309–331, Jun 2017.

[73] Hui Liu, Hong-qi Tian, Yan-fei Li, and Lei Zhang. Comparison of four adaboost algorithm based artificial neural networks in wind speed predictions. *Energy Conversion and Management*, 92:67–81, 2015.

[74] Naisen Liu, Weixing Cao, Yan Zhu, Jingchao Zhang, Fangrong Pang, and Jun Ni. Node deployment with k-connectivity in sensor networks for crop information full coverage monitoring. *Sensors*, 16(12):2096–2119, 2016.

[75] EV Lukina, KW Freeman, KJ Wynn, WE Thomason, RW Mullen, ML Stone, JB Solie, AR Klatt, GV Johnson, RL Elliott, et al. Nitrogen fertilization optimization algorithm based on in-season estimates of yield and plant nitrogen uptake. *Journal of Plant Nutrition*, 24(6):885–898, 2001.

[76] Bruce Maxwell, Paul Hegedus, Philip Davis, Anton Bekkerman, Robert Payn, John Sheppard, Nicholas Silverman, and Clemente Izurieta. Can optimization associated with on-farm experimentation using on-farm experimentation using site-specific technologies improve producer management decisions. In *International Conference on Precision Agriculture (ICPA)*, 2018.

[77] Alex McBratney, Brett Whelan, Tihomir Ancev, and Johan Bouma. Future directions of precision agriculture. *Precision Agriculture*, 6(1):7–23, Feb 2005.

[78] Warren S. McCulloch and Walter Pitts. Neurocomputing: Foundations of research. chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press, Cambridge, MA, USA, 1988.

[79] Mina Mirhosseini, Fatemeh Barani, and Hossein Nezamabadi-pour. Design optimization of wireless sensor networks in precision agriculture using improved bqigsa. *Sustainable Computing: Informatics and Systems*, 16:38–47, 2017.

[80] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[81] Raul Morais, Miguel A Fernandes, Samuel G Matos, Carlos Serôdio, PJSG Ferreira, and MJCS Reis. A zigbee multi-powered wireless acquisition device for remote sensing applications in precision viticulture. *Computers and Electronics in Agriculture*, 62(2):94–106, 2008.

[82] Fabio Moretti, Stefano Pizzuti, Stefano Panzieri, and Mauro Annunziato. Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling. *Neurocomputing*, 167:3–7, 2015.

[83] David J. Mulla. Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps. *Biosystems Engineering*, 114(4):358–371, 2013. Special Issue: Sensing Technologies for Sustainable Agriculture.

[84] Koushik Nagasubramanian, Sarah Jones, Soumik Sarkar, Asheesh K. Singh, and Arti S. Ganapathysubramanian. Hyperspectral band selection using genetic algorithm and support vector machines for early identification of charcoal rot disease in soybean stems. *Plant Methods*, 14:86–99, 2018.

[85] A.H. El Nahry, R.R. Ali, and A.A. El Baroudy. An approach for precision farming under pivot irrigation system using remote sensing and gis techniques. *Agricultural Water Management*, 98(4):517–531, 2011.

[86] Rajathi Natarajan, Jayashree Subramanian, and Elpiniki I. Papageorgiou. Hybrid learning of fuzzy cognitive maps for sugarcane yield classification. *Computers and Electronics in Agriculture*, 127:147–157, 2016.

[87] Joo Camargo Neto, George E. Meyer, and David D. Jones. Individual leaf extractions from young canopy images using gustafsonkessel clustering and a genetic algorithm. *Computers and Electronics in Agriculture*, 51(1):66–85, 2006.

[88] Raimo Nikkil, Ilkka Seilonen, and Kari Koskinen. Software architecture for farm management information systems in precision agriculture. *Computers and Electronics in Agriculture*, 70(2):328–336, 2010. Special issue on Information and Communication Technologies in Bio and Earth Sciences.

[89] Jan Nilsson. Critical loads for sulphur and nitrogen. In *Air pollution and Ecosystems*, pages 85–91. Springer, 1988.

[90] N. Noguchi, J.F. Reid, E.R. Benson, and T.S. Stombaugh. Vision intelligence for an agricultural mobile robot using a neural network. *International Federation of Automatic Control Proceedings Volumes*, 31(5):139–144, 1998.

[91] S.L. Osborne, J.S. Schepers, D.D. Francis, and M.R. Schlemmer. Use of spectral radiance to estimate in-season biomass and grain yield in nitrogen- and waterstressed corn. *Crop Science*, 42:165–171, 2002.

[92] Yakov Pachepsky and Basil Acock. Stochastic imaging of soil parameters to assess variability and uncertainty of crop yield estimates. *Geoderma*, 85(2):213–229, 1998.

[93] Sudhanshu Sekhar Panda, Daniel P Ames, and Suranjan Panigrahi. Application of vegetation indices for agricultural crop yield prediction using neural network techniques. *Remote Sensing*, 2(3):673–696, 2010.

[94] Søren Marcus Pedersen and KM Lind. Precision agriculture–from mapping to site-specific application. In *Precision Agriculture: Technology and Economic Perspectives*, pages 1–20. Springer, 2017.

[95] Amy Peerlinck, John Sheppard, and Bruce Maxwell. Using deep learning in yield and protein prediction of winter wheat based on fertilization prescriptions in precision agriculture. In *International Conference on Precision Agriculture (ICPA)*, 2018.

[96] F.J. Pierce, N.W. Anderson, T.S. Colvin, J.K. Schueller, D.S. Humburg, and N.B. McLaughlin. Yield mapping. In *The Site-Specific Management for Agricultural Systems*, chapter 11, pages 211–243. American Society of Agronomy, Crop Science Society of America, Soil Science Society of America, 677 S. Segoe Rd., Madison, WI 53711, USA, 1997.

[97] Francis J. Pierce and Peter Nowak. Aspects of precision agriculture. volume 67 of *Advances in Agronomy*, pages 1–85. Academic Press, 1999.

[98] Emanuele Pierpaoli, Giacomo Carli, Erika Pignatti, and Maurizio Canavari. Drivers of precision agriculture technologies adoption: A literature review. *Procedia Technology*, 8:61–69, 2013. 6th International Conference on Information and Communication Technologies in Agriculture, Food and Environment (HAICTA 2013).

[99] D. Pokrajac and Z. Obradovic. Neural network-based software for fertilizer optimization in precision farming. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 2110–2115, July 2001.

[100] Anup K Prasad, Lim Chai, Ramesh P Singh, and Menas Kafatos. Crop yield estimation model for iowa using remote sensing and surface parameters. *International Journal of Applied Earth Observation and Geoinformation*, 8(1):26–33, 2006.

[101] Marc' Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 1185–1192, 2007.

[102] J Rasmussen. A model for prediction of yield response in weed harrowing. *Weed Research*, 31(6):401–408, 1991.

[103] William R Raun, John B Solie, Gordon V Johnson, Marvin L Stone, Robert W Mullen, Kyle W Freeman, Wade E Thomason, and Erna V Lukina. Improving nitrogen use efficiency in cereal grain production with optical sensing and variable rate application. *Agronomy Journal*, 94(4):815–820, 2002.

[104] JA López Riquelme, Fulgencio Soto, J Suardíaz, P Sánchez, Andres Iborra, and JA Vera. Wireless sensor networks for precision horticulture in southern spain. *Computers and Electronics in Agriculture*, 68(1):25–35, 2009.

[105] P. C. Robert. *Precision agriculture: a challenge for crop nutrition management*, pages 143–149. Springer Netherlands, Dordrecht, 2002.

[106] RA Viscarra Rossel and AB McBratney. Soil chemical analytical accuracy and costs: implications from precision agriculture. *Australian Journal of Experimental Agriculture*, 38(7):765–775, 1998.

[107] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 322:533–536, October 1986.

[108] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. In James A. Anderson and Edward

Rosenfeld, editors, *Neurocomputing: Foundations of Research*, pages 696–699. MIT Press, Cambridge, MA, 1988.

[109] Helena Russello. Convolutional neural networks for crop yield prediction using satellite images. Master's thesis, University of Amsterdam. Artificial Intelligence.

[110] Georg Ru and Alexander Brenning. Spatial variable importance assessment for yield prediction in precision agriculture. In Paul R. Cohen, Niall M. Adams, and Michael R. Berthold, editors, *Advances in Intelligent Data Analysis IX*, pages 184–195, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[111] Georg Ru, Rudolf Kruse, Martin Schneider, and Peter Wagner. Data mining with neural networks for wheat yield prediction. In *Lecture Notes in Computer Science*, volume 5077, pages 47–56, 2008.

[112] Inkyu Sa, Zongyuan Ge, Feras Dayoub, Ben Upcroft, Tristan Perez, and Chris McCool. Deepfruits: A fruit detection system using deep neural networks. *Sensors*, 16(8), 2016.

[113] A. Izabela Samborska, Vladimir Alexandrov, Leszek Sieczko, Boena Kornatowska, Vasilij Goltsev, Magdalena D. Cetner, and Hazem M. Kalaji. Artificial neural networks and their application in biological and agricultural research. *Signpost Open Access Journal of NanoPhotoBioSciences*, 2:14–30, 2014.

[114] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[115] Holger Schwenk and Yoshua Bengio. Training methods for adaptive boosting of neural networks. In *Advances in Neural Information Processing Systems*, pages 647–653, 1998.

[116] Holger Schwenk and Yoshua Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000.

[117] G.A.F. Seber and A.J. Lee. *Linear Regression Analysis*. Wiley, 2012.

[118] Dimitri P Solomatine and Durga L Shrestha. Adaboost. rt: a boosting algorithm for regression problems. *Neural Networks*, 2:1163–1168, 2004.

[119] CJT Spitters, MJ Kropff, and W De Groot. Competition between maize and echinochloa crus-galli analysed by a hyperbolic regression model. *Annals of Applied Biology*, 115(3):541–551, 1989.

[120] John Stafford. Implementing precision agriculture in the 21st century. *Journal of Agricultural Engineering Research*, 76:267–275, 2000.

[121] Chun Chet Tan and C. Eswaran. Performance comparison of three types of autoencoder neural networks. In *Second Asia International Conference on Modeling & Simulation*, pages 213–218, May 2008.

[122] Lie Tang, L Tian, and Brian L Steward. Color image segmentation with genetic algorithm for in-field weed sensing. *Transactions of the American Society of Agricultural Engineers*, 43(4):1019, 2000.

[123] Dennis Timlin, Yakov Pachepsky, Charles Walthall, and Sara Loechel. The use of a water budget model and yield maps to characterize water availability in a landscape. *Soil Tillage Research*, 58(3):219–231, 2001.

[124] Y Uno, S.O. Prashera, R. Lacroixb, P.K. Goela, Y. Karimia, A. Viauc, and Patel R.M. Artificial neural networks to predict corn yield from compact airborne spectrographic imager data. *Computers and Electronics in Agriculture*, 47:149–161, 2005.

[125] B.J. Van Alphen and J.J. Stoorvogel. A methodology for precision nitrogen fertilization in high-input farming systems. *Precision Agriculture*, 2:319–332, 2000.

[126] Jos Luis Velasco, Hernn Sainz Rozas, Hernn Eduardo Echeverra, and Pablo Andrs Barbieri. Optimizing fertilizer nitrogen use efficiency by intensively managed spring wheat in humid regions: Effect of split application. *Canadian Journal of Plant Science*, 92:847–856, 2012.

[127] Neil Walton, John Sheppard, and Joseph Shaw. Genetic wavelength selection and filter specification for multi-spectral image classification. In *submitted to Genetic and Evolutionary Computation Conference*, 2019.

[128] K. Bradley Watkins, Yao chi Lu, and Wen yuan Huang. Economic and environmental feasibility of variable rate nitrogen fertilizer application with carry-over effects. *Journal of Agricultural and Resource Economics*, 23(2):401–426, 1998.

[129] Paul Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974. Applied Mathematics.

[130] K. M. Whitley, J. R. Davenport, and S. R. Manley. Differences in nitrate leaching under variable and conventional nitrogen fertilizer management in irrigated potato systems. In P. C. Robert, R. H. Rust, and W. E. Larson, editors, *Proceedings of the 5th International Conference on Precision Agriculture*, pages 1–9, Madison, USA, 2000. American Society of Agronomy.

[131] Luo Xiwen, Ou Yinggang, Zhao Zuoxi, Zhao Xin, Zhang Zhigang, Zhang Zhibin, Lu Guangyu, and Li Junling. Research and development of intelligent flexible chassis for precision farming [j]. *Transactions of The Chinese Society of Agricultural Engineering*, 2:0–19, 2005.

[132] Jiaxuan You, Xiaocheng Li, Melvin Low, David Lobell, and Stefano Ermon. Deep gaussian process for crop yield prediction based on remote sensing data. In *Association for the Advancement of Artificial Intelligence*, pages 4559–4566, 2017.

[133] Helong Yu, Dayou Liu, Guifen Chen, Baocheng Wan, Shengsheng Wang, and Bo Yang. A neural network ensemble method for precision fertilization modeling. *Mathematical and Computer Modelling*, 51:1375–1382, 2010.

[134] Naiqian Zhang, Maohua Wang, and Ning Wang. Precision agriculture: a worldwide overview. *Computers and Electronics in Agriculture*, 36:113–132, 2002.

[135] Peng-Bo Zhang and Zhi-Xin Yang. A novel adaboost framework with robust threshold and structural optimization. *IEEE Transactions on Cybernetics*, 2016.

[136] Yu-Jun Zheng, Qin Song, and Sheng-Yong Chen. Multiobjective fireworks optimization for variable-rate fertilization in oil crop production. *Applied Soft Computing*, 13(11):4253–4263, 2013.

[137] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 137(1-2):239–263, 2002.

APPENDICES

APPENDIX A

APPLICATION TOOL SCREENSHOTS

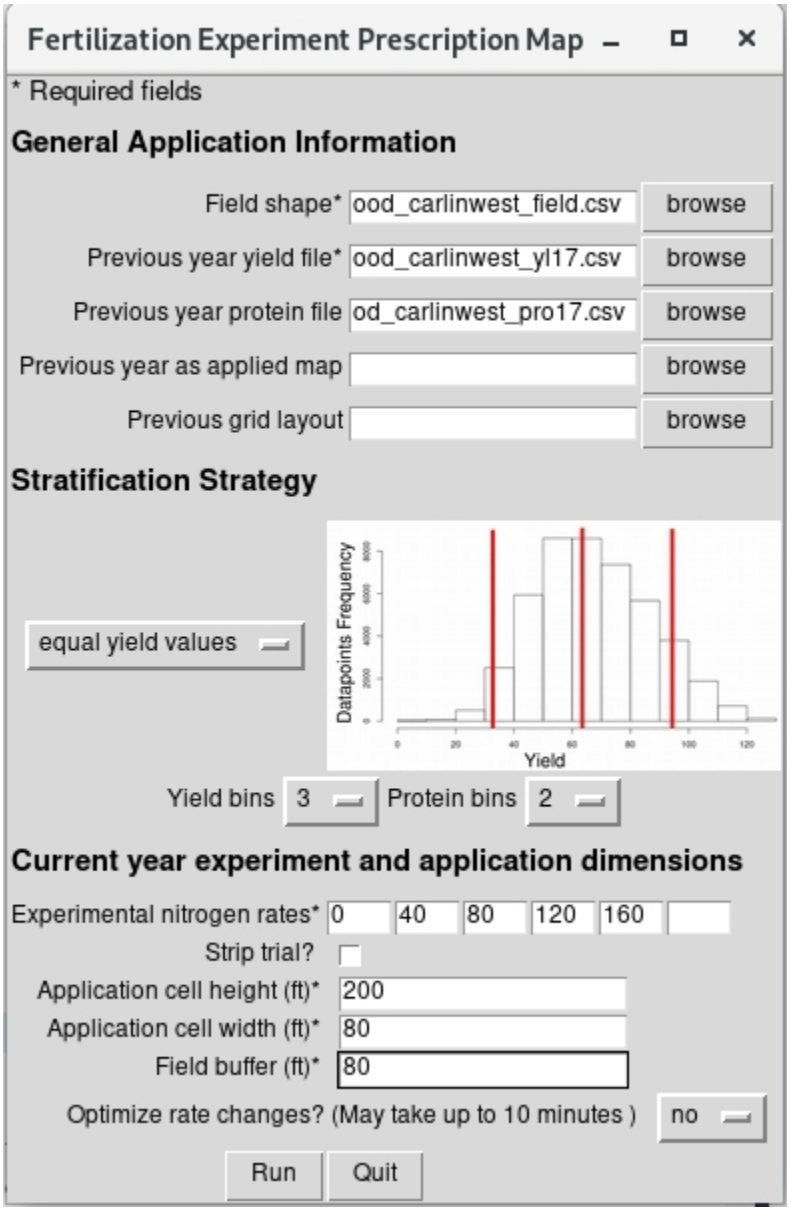


Figure A.1: Screenshot showing the main user input interface for the prescription map generation tool.

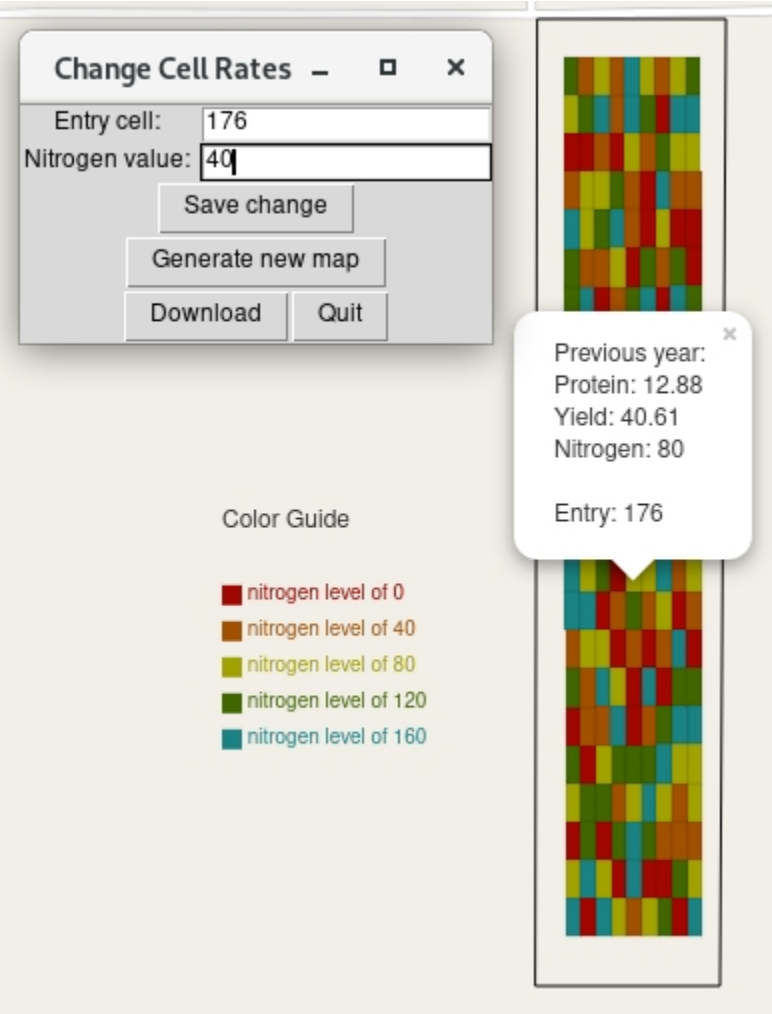


Figure A.2: Screenshot showing the ability to adjust a single cell's nitrogen value. This screen also shows how to generate a new map after the desired changes have been made, or how to download the current map as it is. The field shown is sre1314.

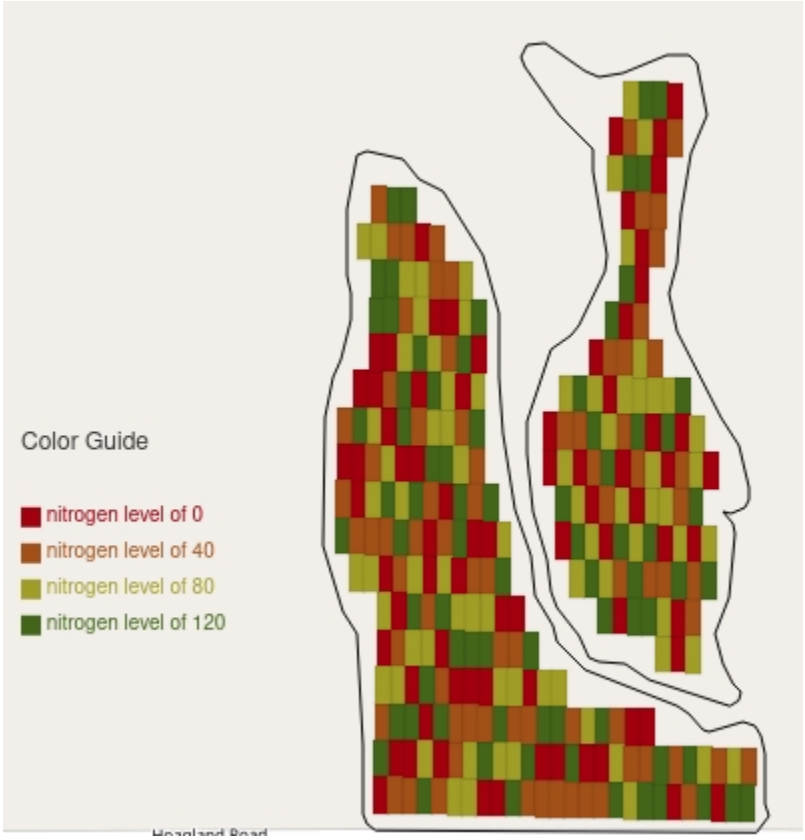


Figure A.3: Screenshot showing the prescription map as it pops up in the browser and the corresponding legend for field sec35mid.

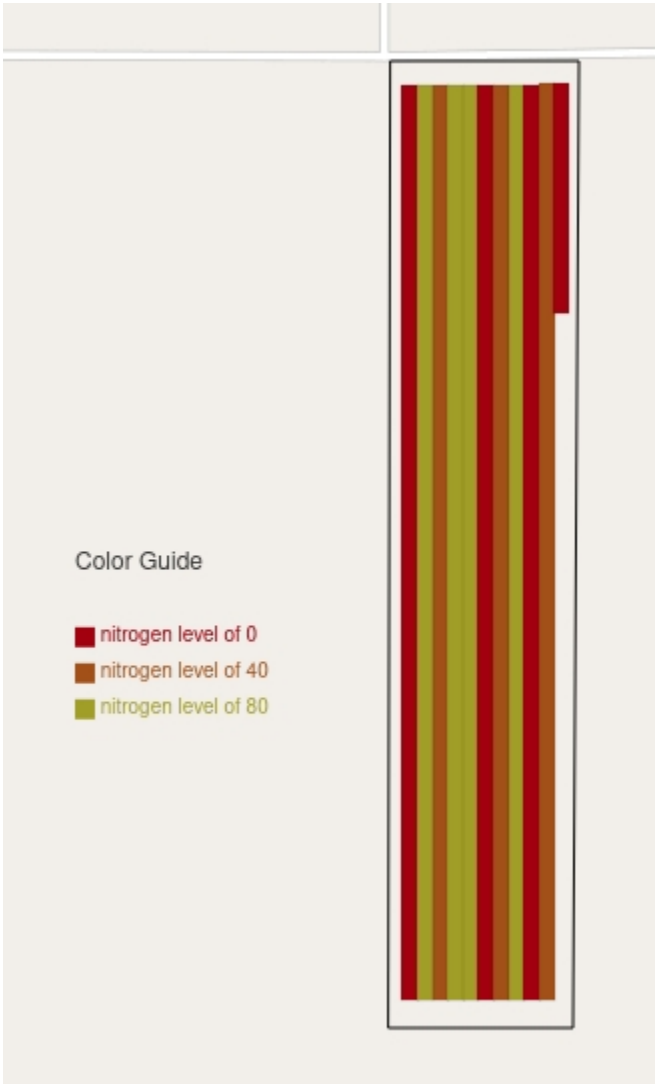


Figure A.4: Example of prescription map using a strip trial layout for field sre1314.

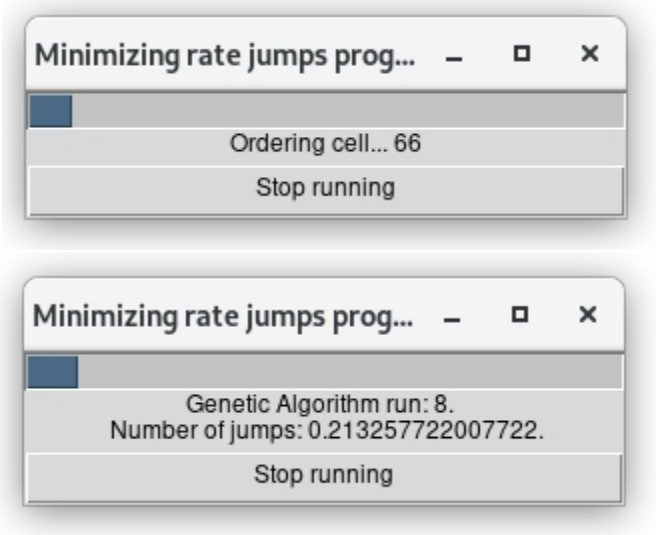
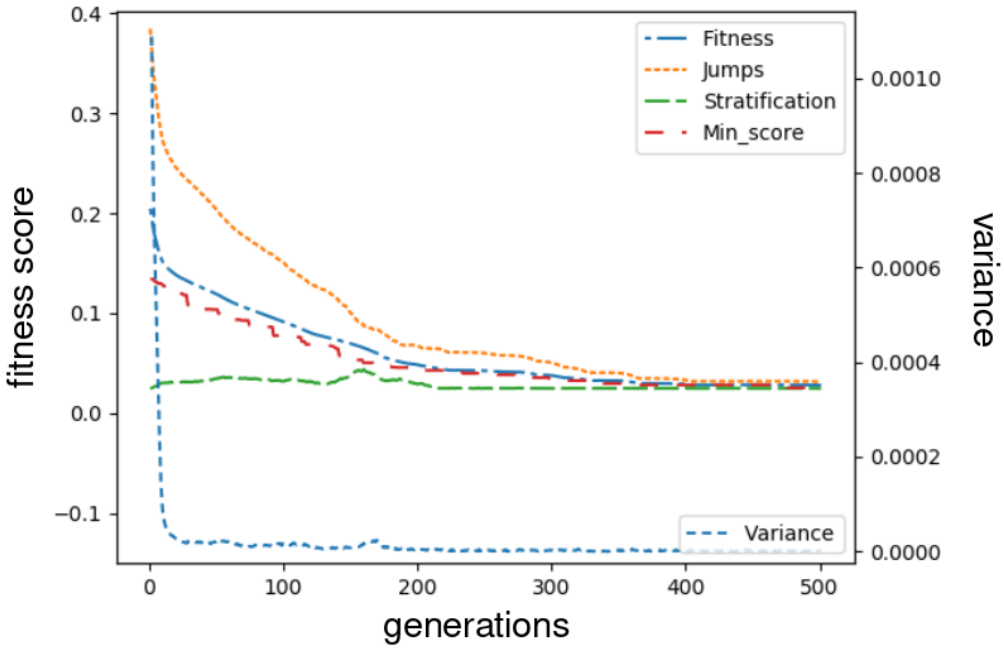


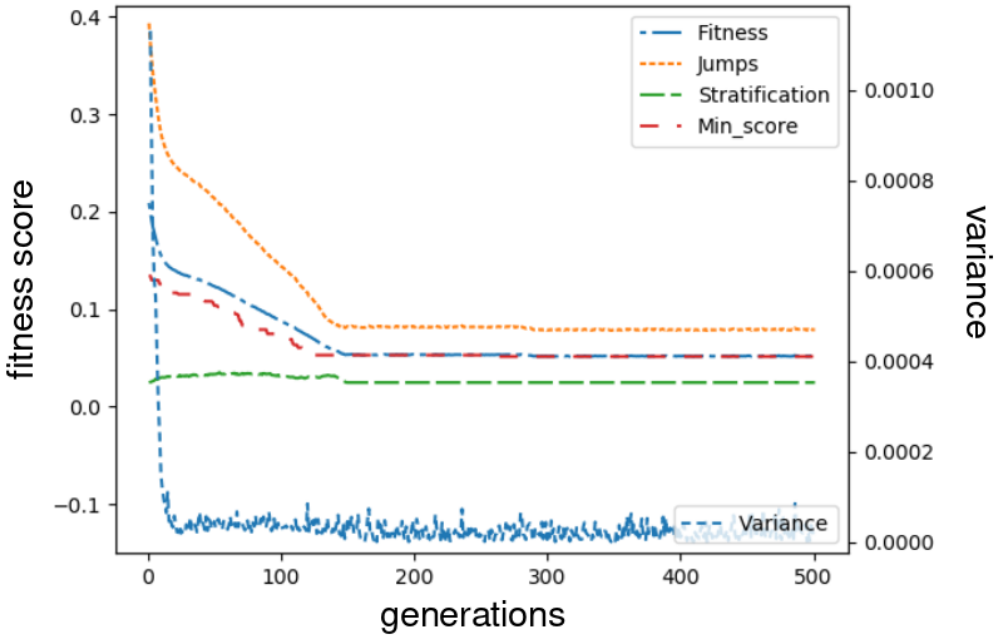
Figure A.5: The progress bar that shows up when the genetic algorithm is running. The progress bar shows two stages: the ordering of the cells and the actual genetic algorithm. The pop-up also includes the ability to stop the genetic algorithm prematurely.

APPENDIX B

GENETIC ALGORITHM GENERATIONAL PLOTS

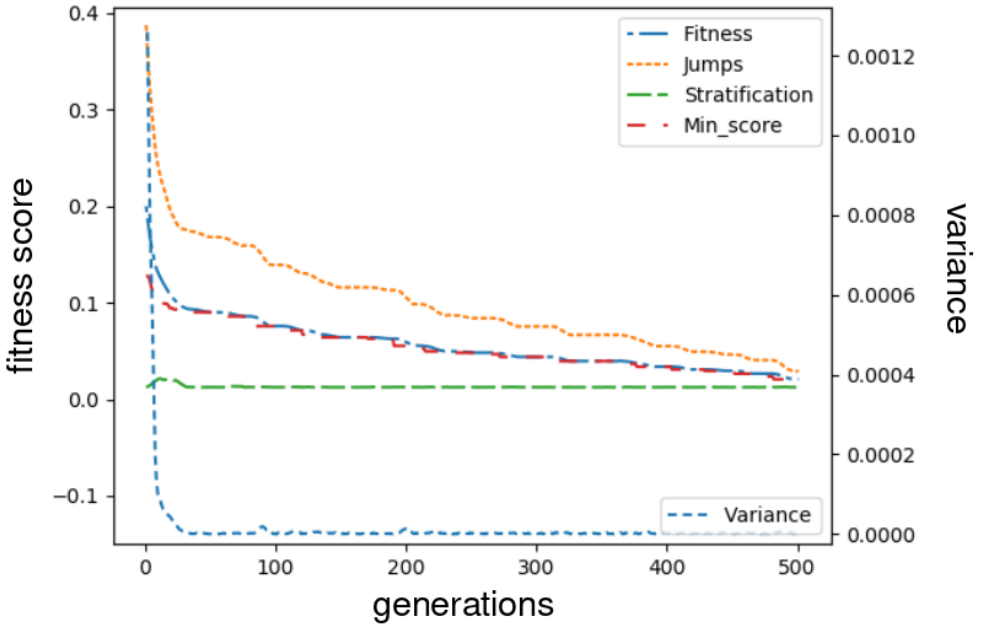


(a) Swap mutation using tournament size 3.

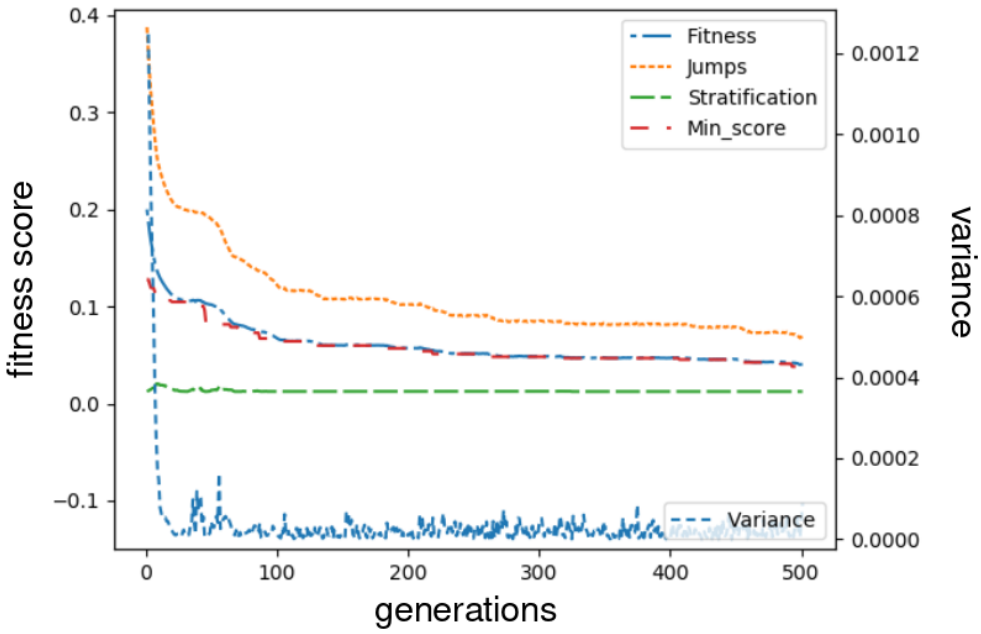


(b) Scramble mutation using tournament size 3.

Figure B.1: Sec35mid results for 500 generations of the GA using the two different mutation types and equal sample binning, where $w = 0.5$. The left y-axis shows the fitness score values (including jumps, stratification and the best score in the population), while the right y-axis details the variance value.

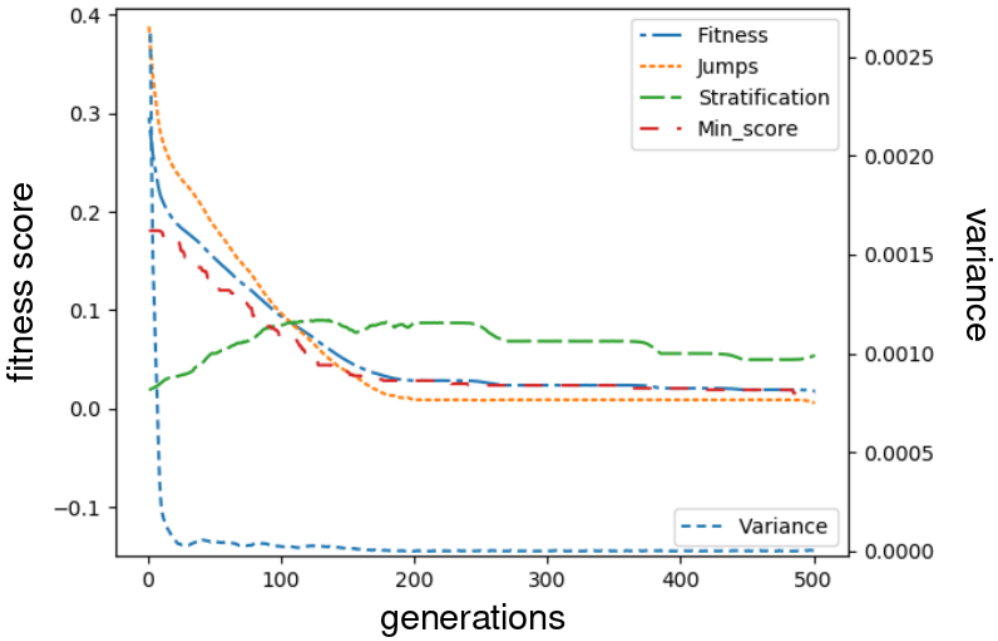


(a) Swap mutation using tournament size 20.

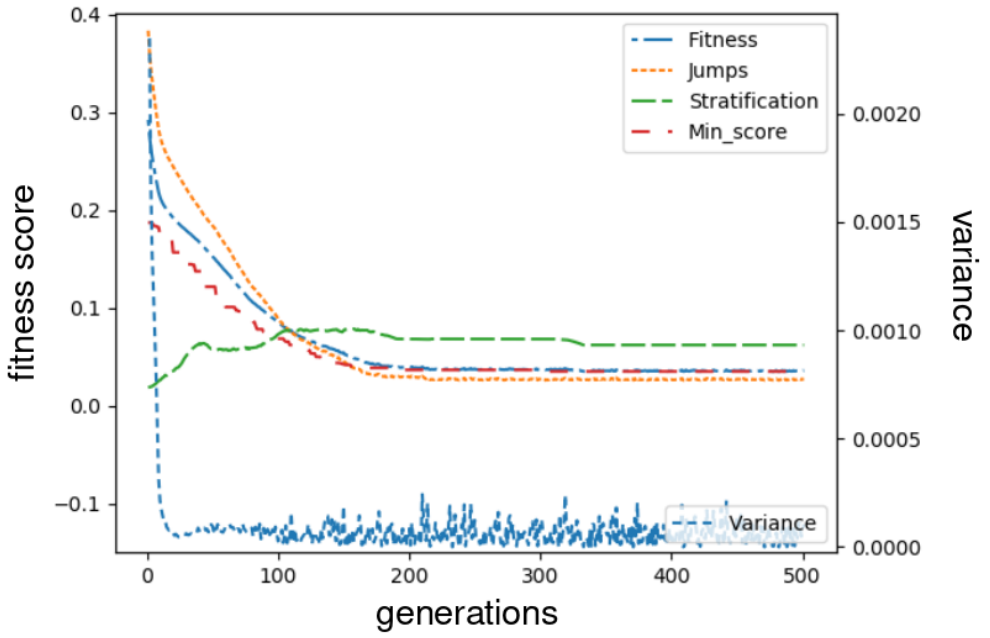


(b) Scramble mutation using tournament size 20.

Figure B.2: Sec35mid results for 500 generations of the GA using the two different mutation types and equal sample binning, where $w = 0.5$. The left y-axis shows the fitness score values (including jumps, stratification and the best score in the population), while the right y-axis details the variance value.

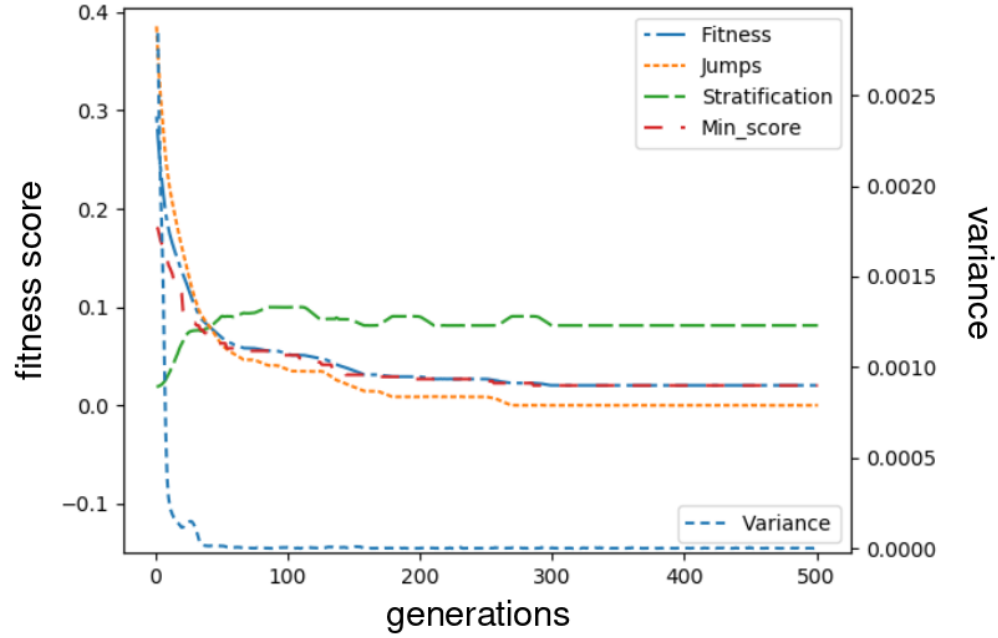


(a) Swap mutation using tournament size 3.

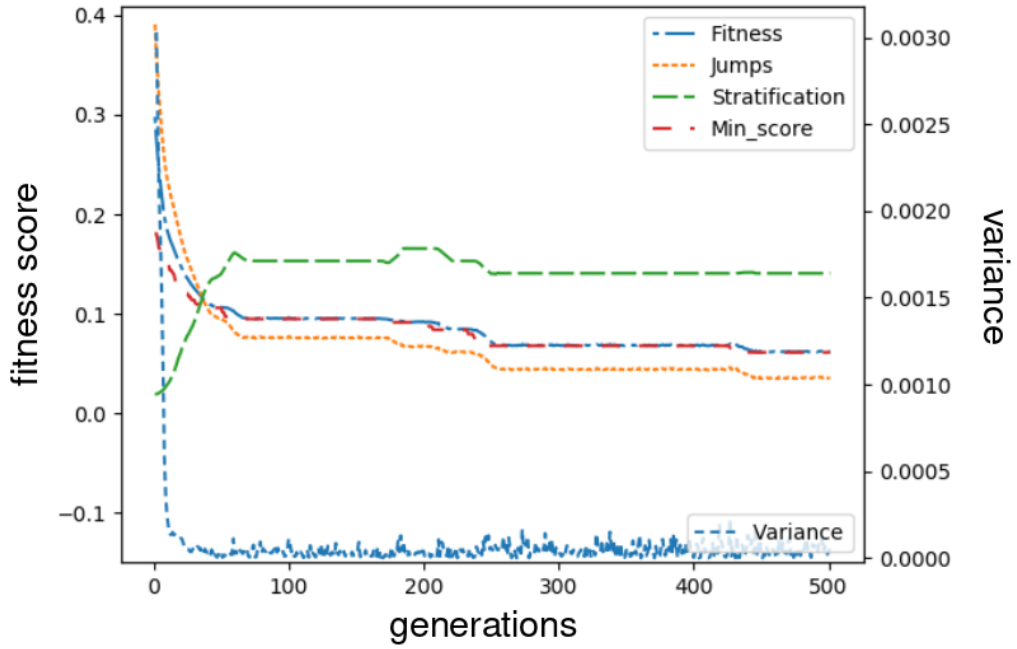


(b) Scramble mutation using tournament size 3.

Figure B.3: Sec35mid results for 500 generations of the GA using the two different mutation types and the two different discretization methods, where $w = 0.75$. The left y-axis shows the fitness score values (including jumps, stratification, and the best score in the population), while the right y-axis details the variance value.

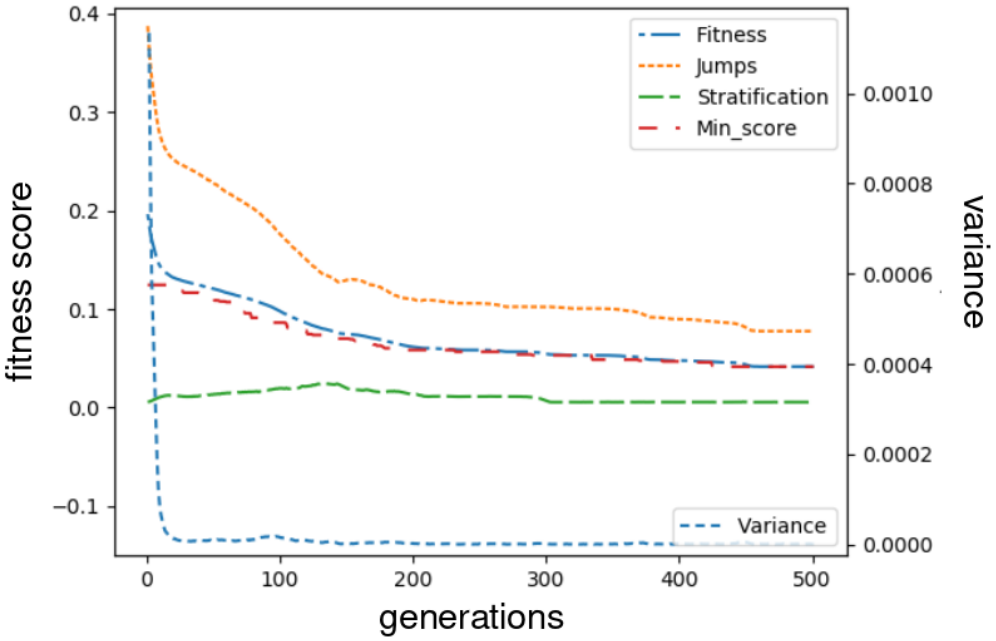


(a) Swap mutation using tournament size 20.

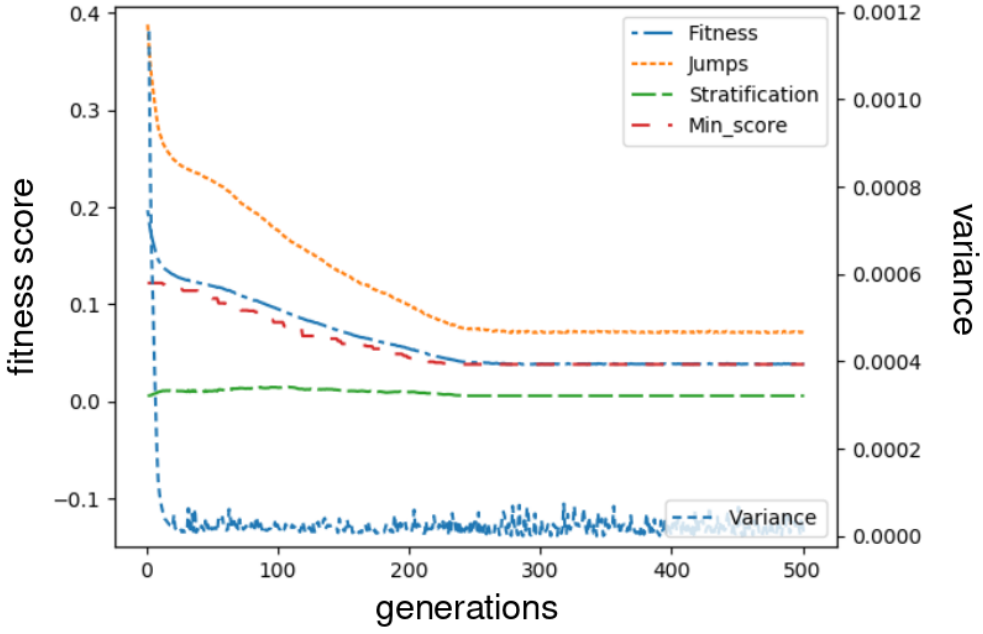


(b) Scramble mutation using tournament size 20.

Figure B.4: Sec35mid results for 500 generations of the GA using the two different mutation types and the two different discretization methods, where $w = 0.75$. The left y-axis shows the fitness score values (including jumps, stratification, and the best score in the population), while the right y-axis details the variance value.

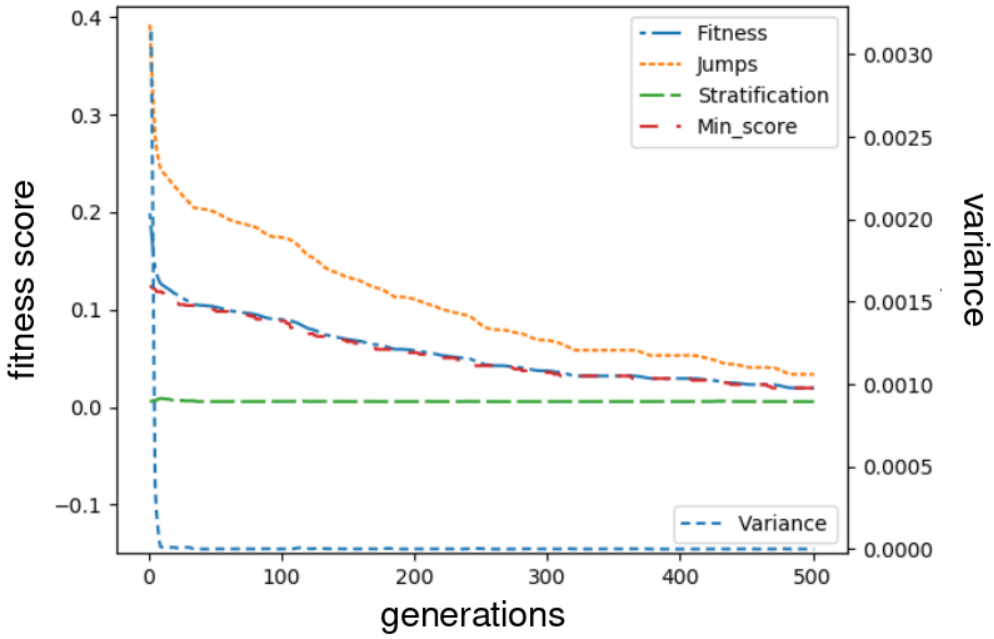


(a) Swap mutation using tournament size 3.

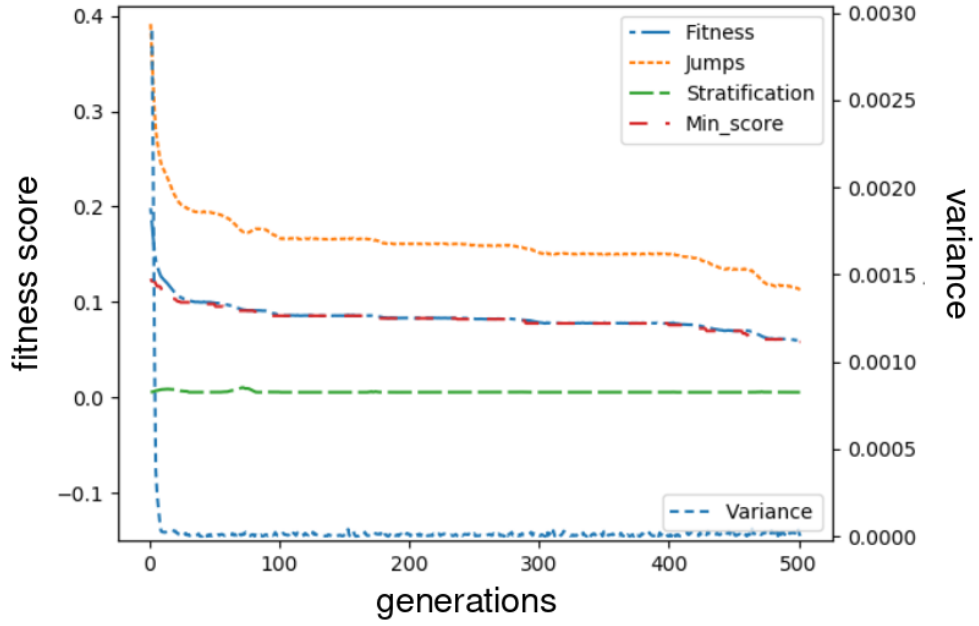


(b) Scramble mutation using tournament size 3.

Figure B.5: Davidsonmidwest results for 500 generations of the GA using the two different mutation types and equal sample binning, where $w = 0.5$. The left y-axis shows the fitness score values (including jumps, stratification and the best score in the population), while the right y-axis details the variance value.

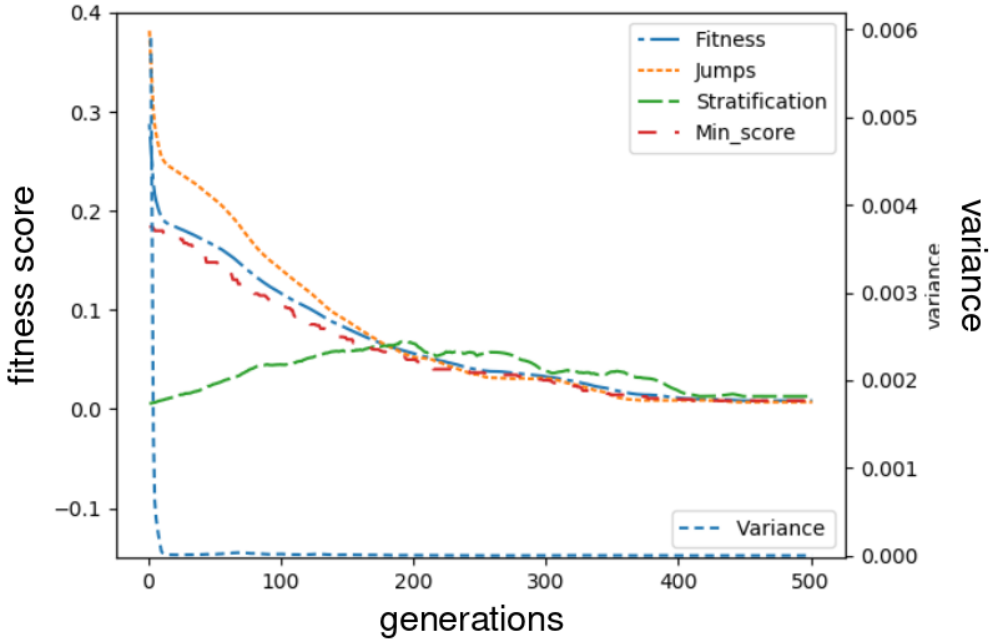


(a) Swap mutation using tournament size 20.

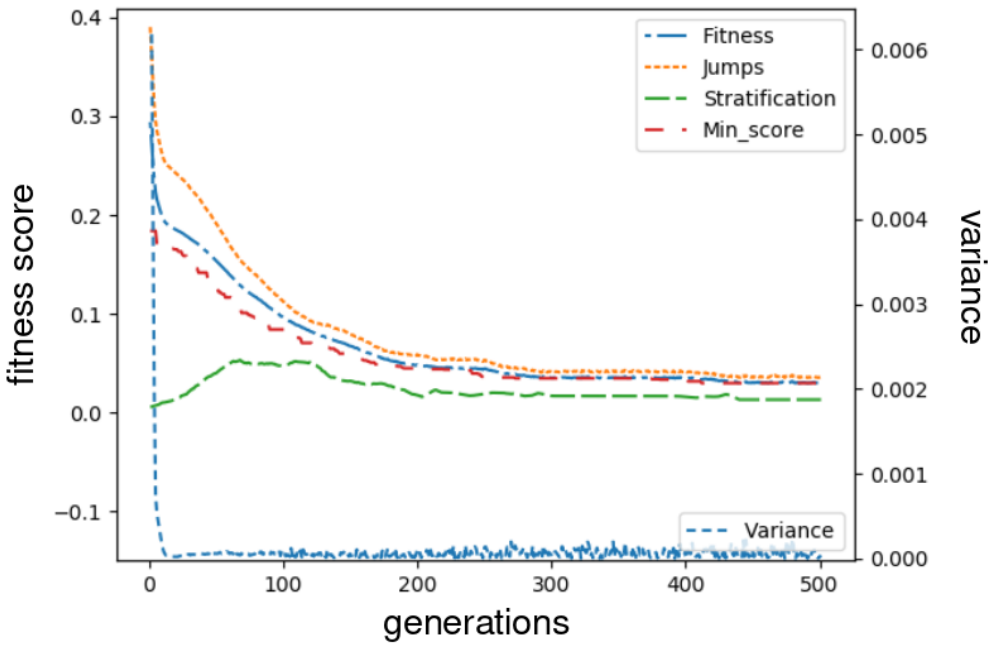


(b) Scramble mutation using tournament size 20.

Figure B.6: Davidsonmidwest results for 500 generations of the GA using the two different mutation types and equal sample binning, where $w = 0.5$. The left y-axis shows the fitness score values (including jumps, stratification and the best score in the population), while the right y-axis details the variance value.

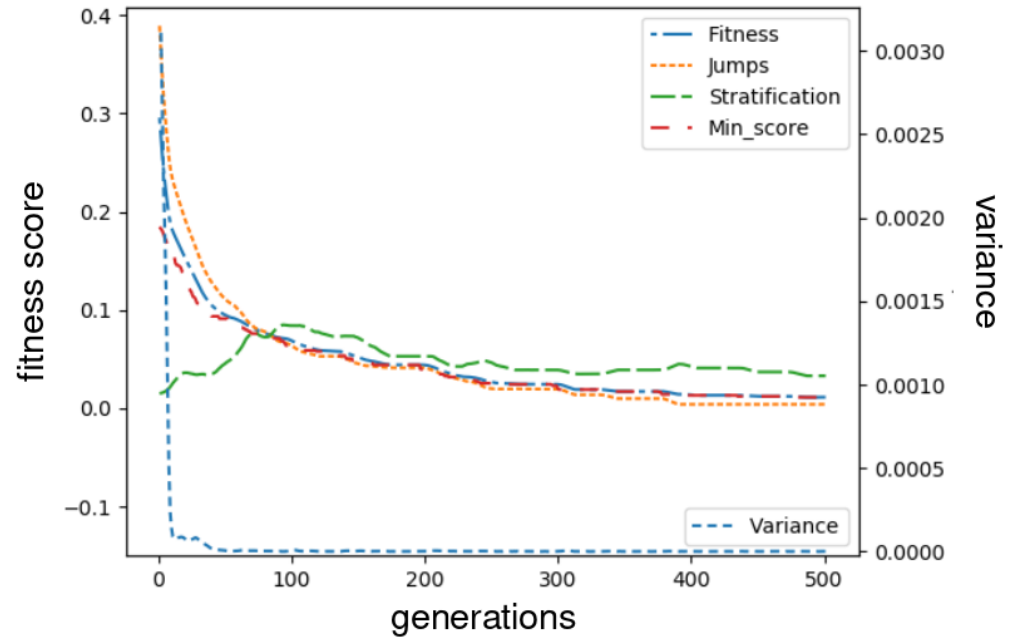


(a) Swap mutation using tournament size 3.

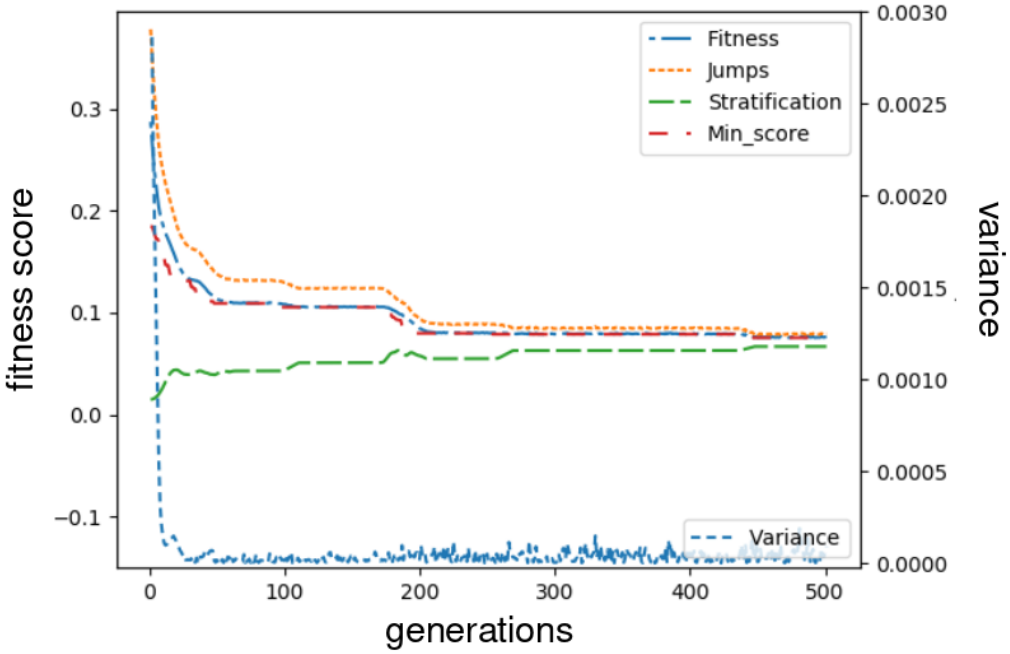


(b) Scramble mutation using tournament size 3.

Figure B.7: Davidsonmidwest results for 500 generations of the GA using the two different mutation types and the two different discretization methods, where $w = 0.75$. The left y-axis shows the fitness score values (including jumps, stratification, and the best score in the population), while the right y-axis details the variance value.



(a) Swap mutation using tournament size 20.

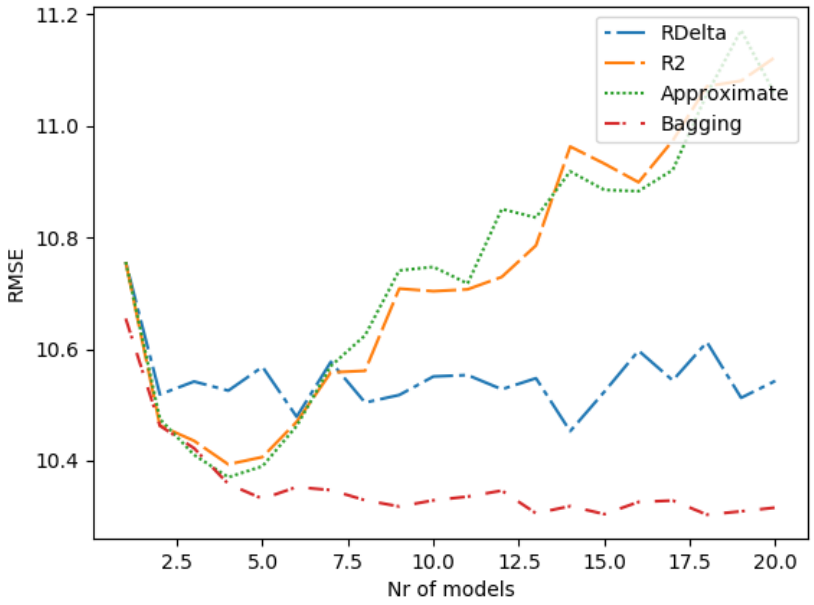


(b) Scramble mutation using tournament size 20.

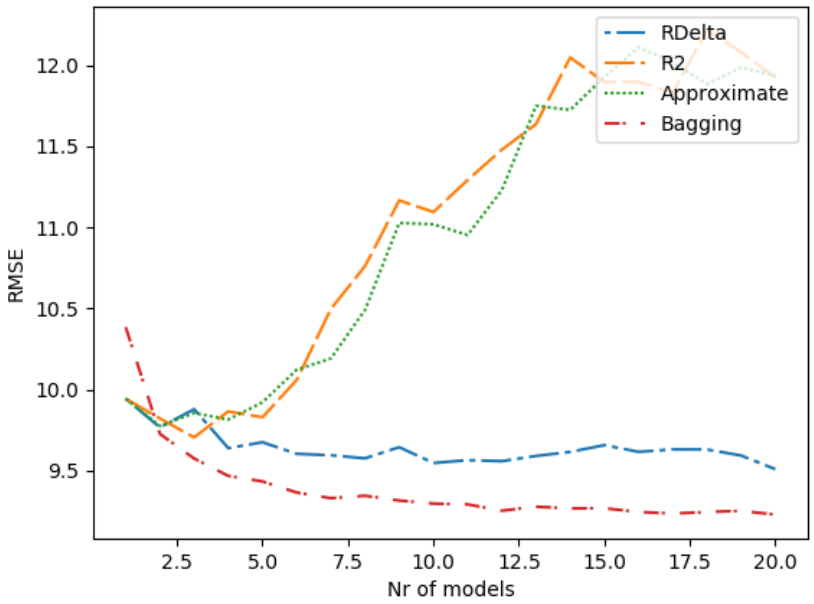
Figure B.8: Davidsonmidwest results for 500 generations of the GA using the two different mutation types and the two different discretization methods, where $w = 0.75$. The left y-axis shows the fitness score values (including jumps, stratification, and the best score in the population), while the right y-axis details the variance value.

APPENDIX C

ENSEMBLE METHOD RMSE PLOTS

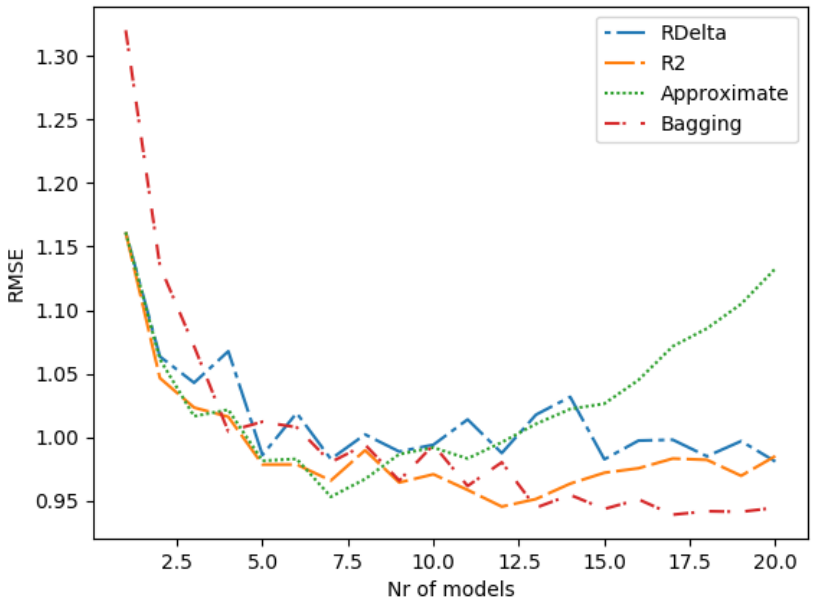


(a) Yield data.

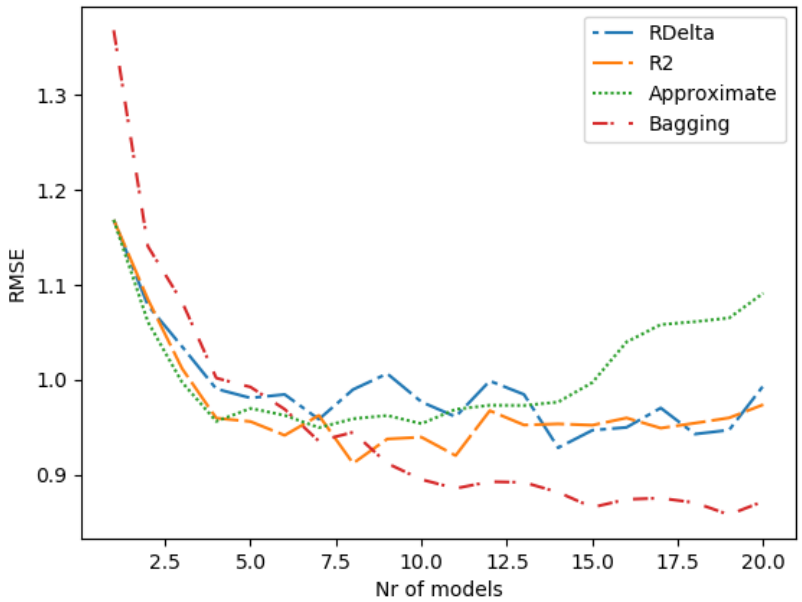


(b) Spatial yield data.

Figure C.1: Ensemble results for "sre1314" yield.

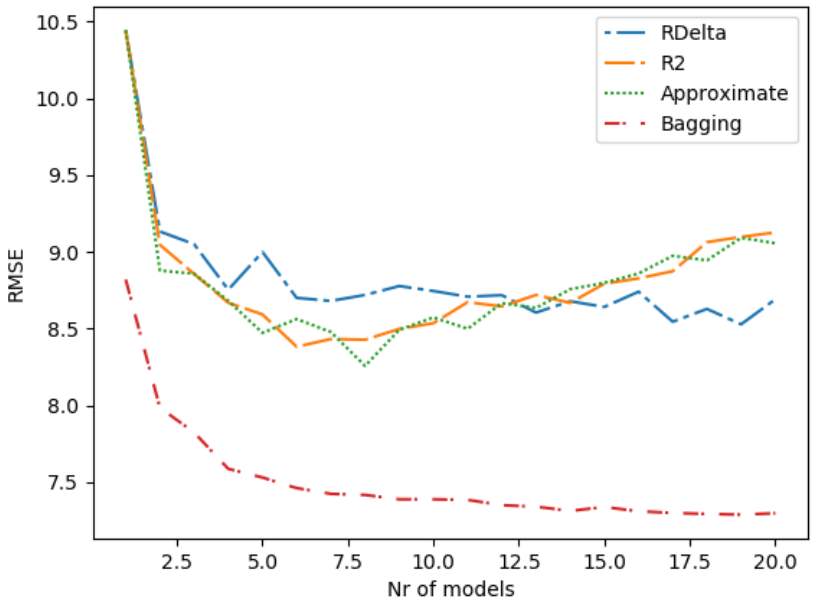


(a) Protein data.

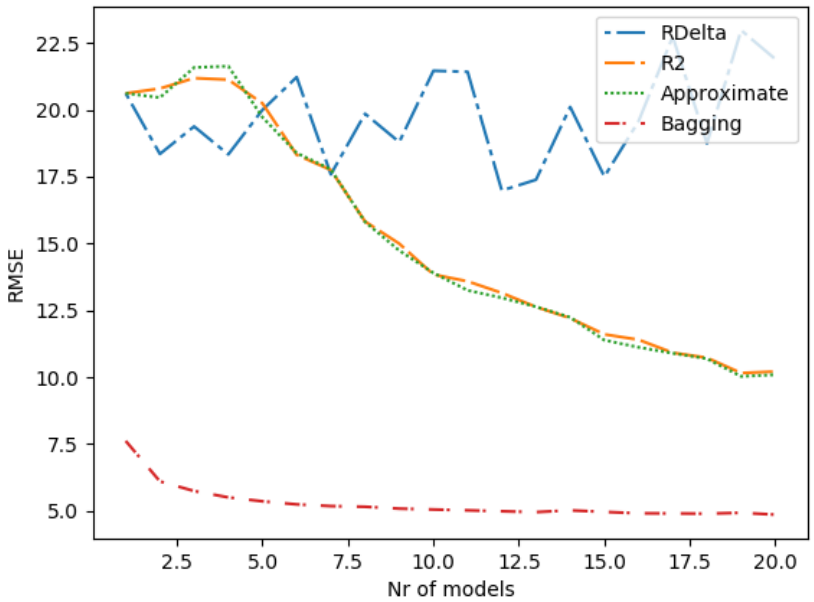


(b) Spatial protein data.

Figure C.2: Ensemble results for "sre1314" protein.

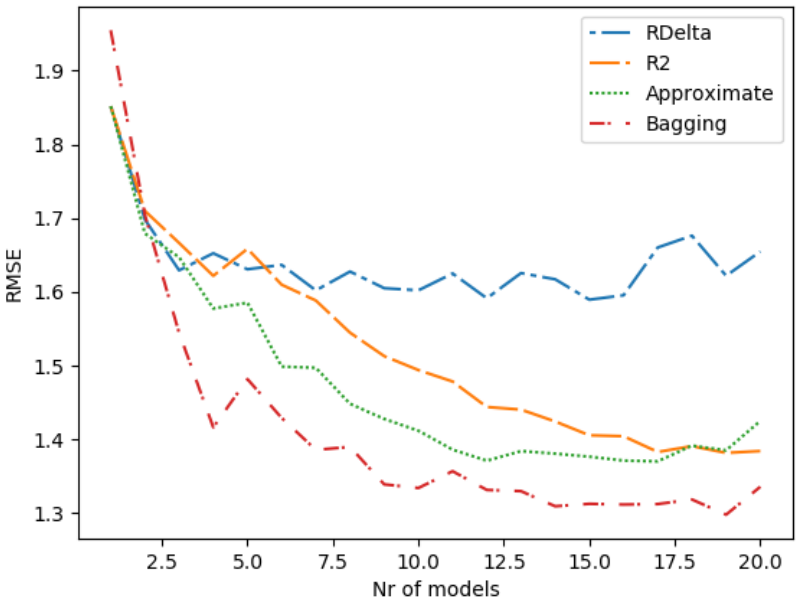


(a) Yield data.

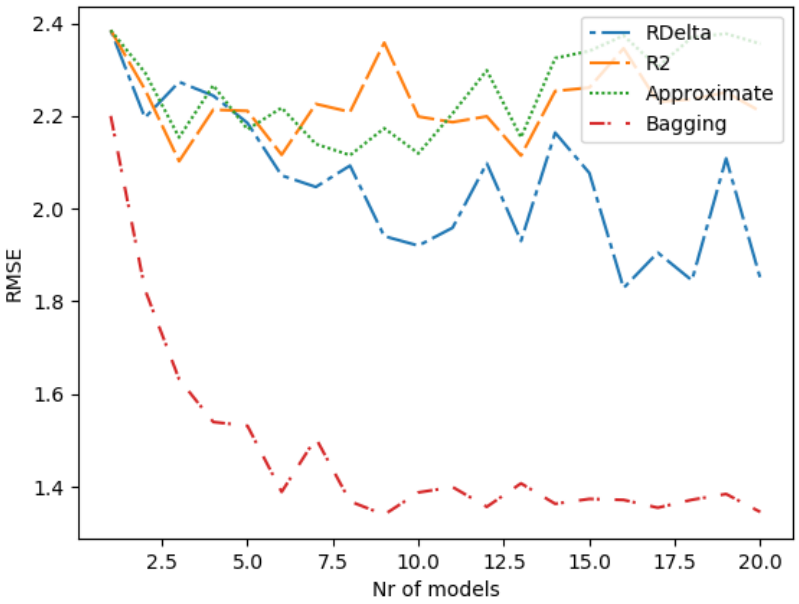


(b) Spatial yield data.

Figure C.3: Ensemble results for "sec35mid" yield.

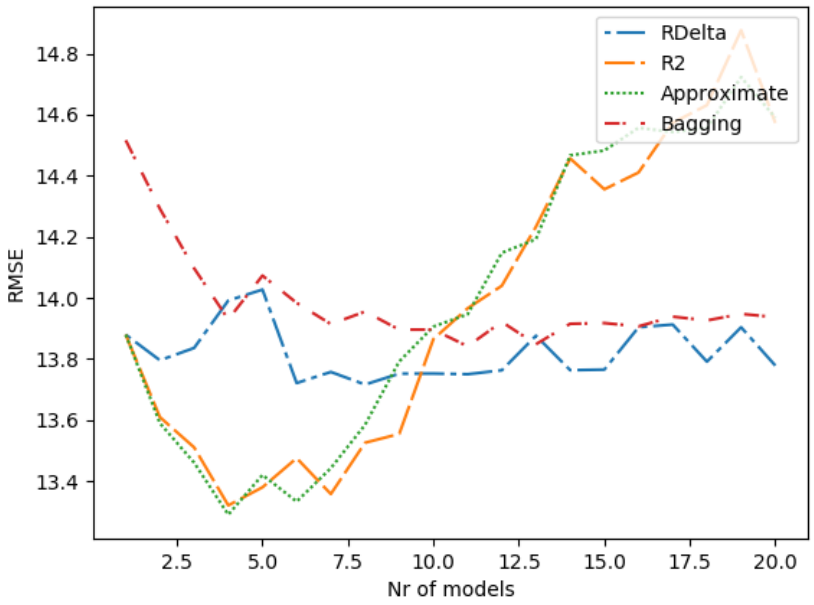


(a) Protein data.

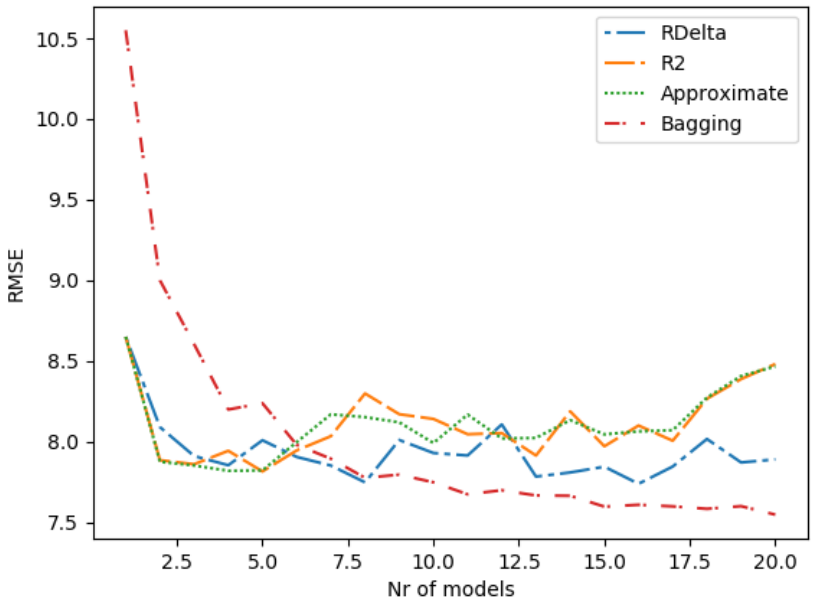


(b) Spatial protein data.

Figure C.4: Ensemble results for "sec35mid" protein.

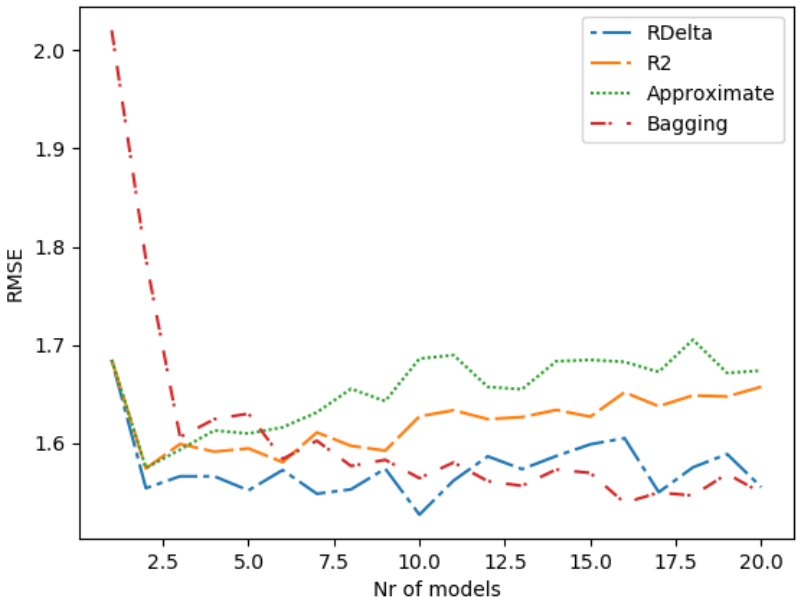


(a) Yield data.

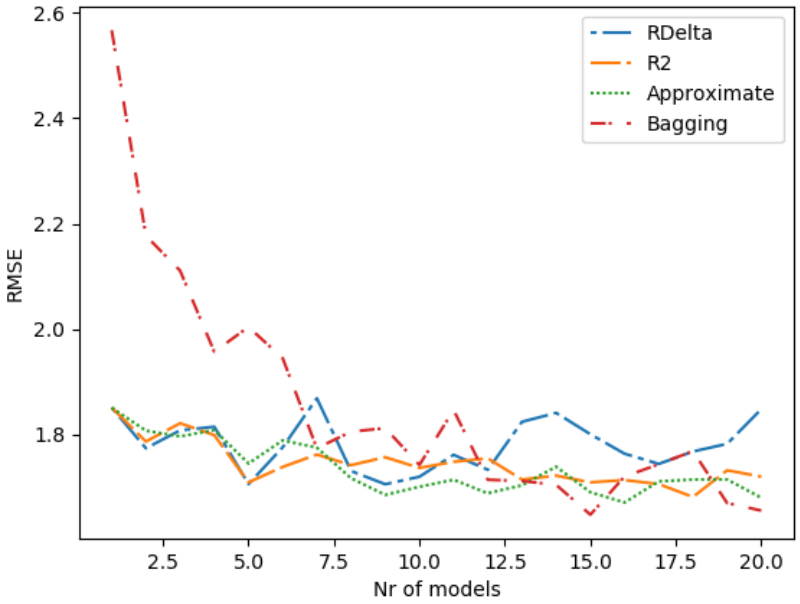


(b) Spatial yield data.

Figure C.5: Ensemble results for "davidsonmidwest" yield.

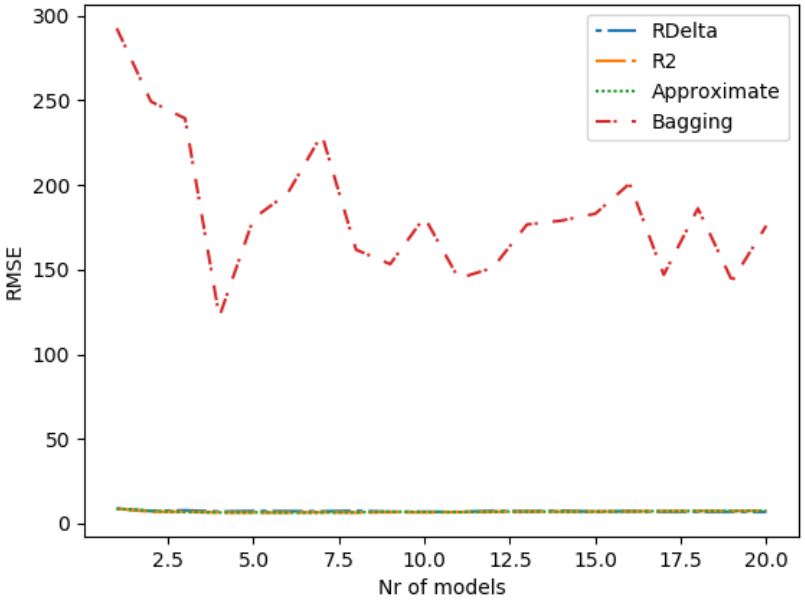


(a) Protein data.

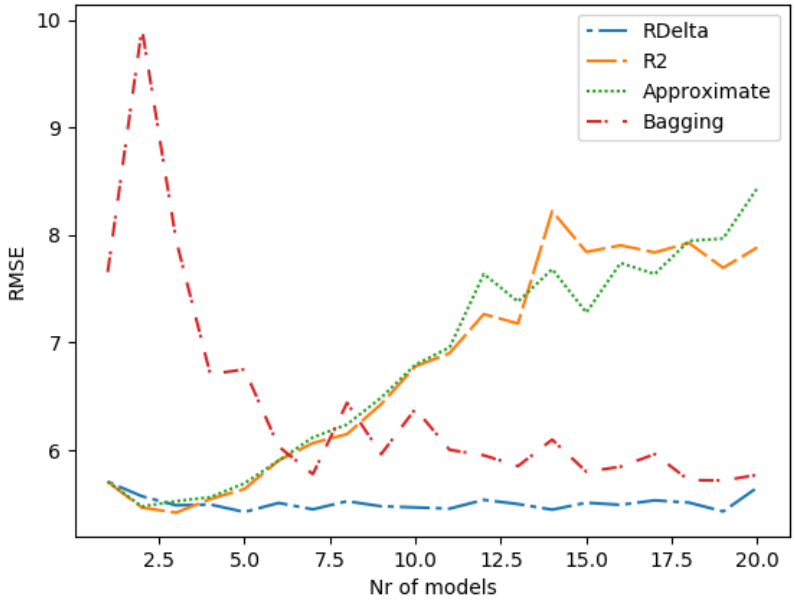


(b) Spatial protein data.

Figure C.6: Ensemble results for "davidsonmidwest" protein.



(a) Yield data.



(b) Spatial yield data.

Figure C.7: Ensemble results for "carlinwest" yield.

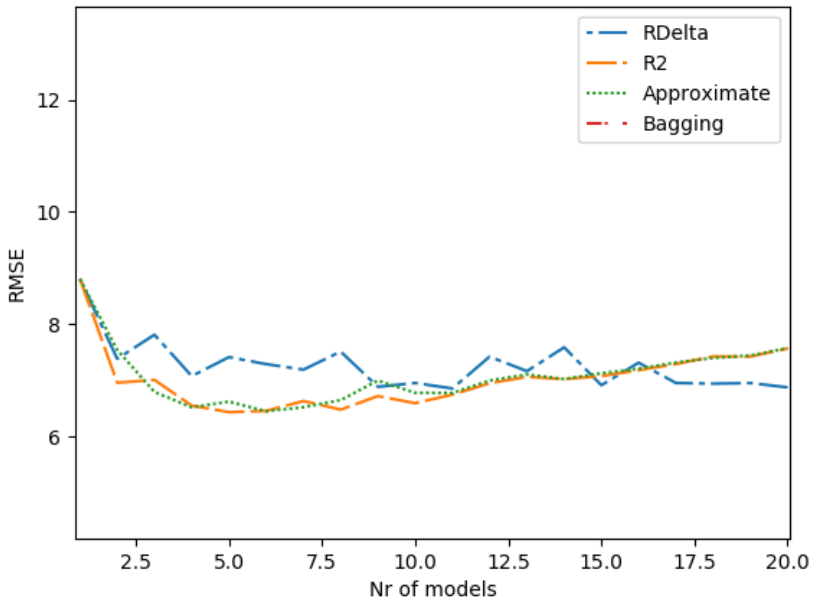
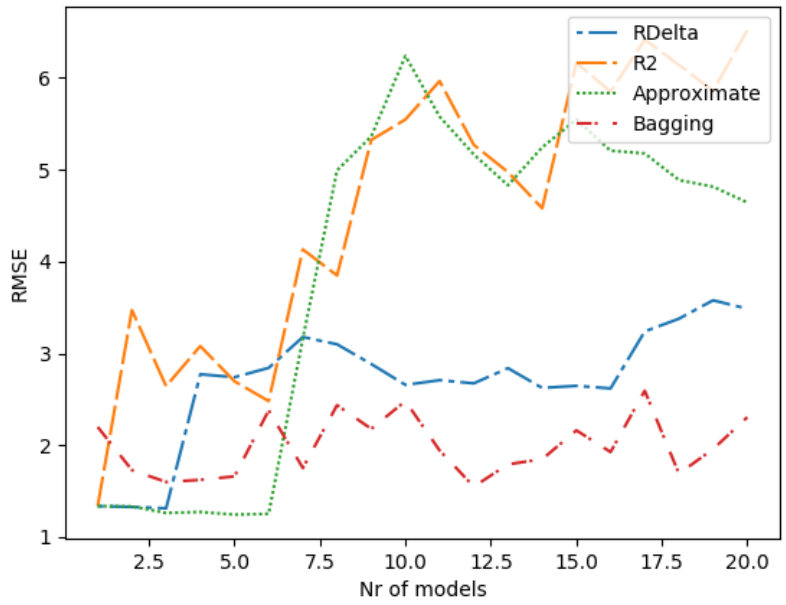
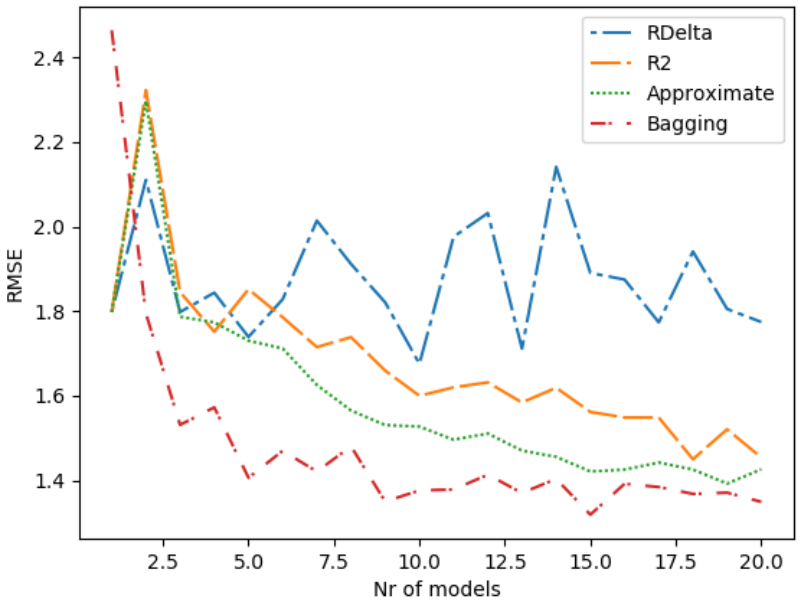


Figure C.8: Spatial yield data zoomed in on boosting models for "carlinwest".



(a) Protein data.



(b) Spatial protein data.

Figure C.9: Ensemble results for "carlinwest" protein.