Disease Prediction using Machine Learning

A Synopsis Submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal
Towards Partial Fulfillment for the Award of

Bachelor of Technology
(Computer Science and Engineering)

Under the Supervision of
Professor Preeti Shukla

Submitted By
Aman Patel (0827CS213D03)
Bhomik Gahlot (0827CS213D08)
Chandransh
chouhan(0827CS201062)
Anurag yadav (0827CS201039)
Anurag Pawar (0827CS201038)

Department of Computer Science and Engineering
Acropolis Institute of Technology & Research, Indore
Jan-June 2023

EXAMINER APPROVAL

The Project entitled "**Disease Prediction using Machine Learning**" submitted by Aman Patel (0827CS213D03), Anurag Yadav (0827CS201039), Bhomik Gahlot (0827CS213D08) and Chandransh Singh Chouhan (0827CS201062) has been examined and is hereby approved towards partial fulfillment for the award of Bachelor of Engineering degree in Computer Science &amp; Engineering discipline, for which it has been submitted. It understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it has been submitted.

(Internal Examiner)	**(External Examiner)**
Date:	**Date:**

GUIDE RECOMMENDATION

This is to certify that the work embodied in this project entitled "**Disease Prediction using Machine Learning**" submitted by Aman Patel (0827CS213D03), Anurag Yadav (0827CS201039), Bhomik Gahlot (0827CS213D08) and Chandransh Singh Chouhan (0827CS201062)Anurag Pawar (0827CS201038) is a satisfactory account of the bonafide work done under the supervision of Prof. Ronak Jain recommended towards partial fulfillment for the award of the Bachelor of Engineering (Computer Science & Engineering) degree by Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal.

(Project Guide) **(Project Coordinator)**

STUDENTS UNDERTAKING

This is to certify that a project entitled “Smart Accessibility Map” has developed by us under the supervision of Prof. Ronak Jain. The whole responsibility of work done in this project is ours. The sole intention of this work is only for practical learning and research. We further declare that to the best of our knowledge this report does not contain any part of any work which has been submitted for the award of any degree either in this University or in any other University / Deemed University without proper citation and if the same work is found then we are liable for explanation to this.

Aman Patel (0827CS213D03) Anurag Yadav (0827CS201039) BhomikGahlot (0827CS213D08) Chandransh Singh Chouhan (0827CS201062)Anurag Pawar(0827CS201038)

Acknowledgement

We thank the almighty Lord for giving me the strength and courage to sail out through the tough and reach on shore safely. There are a number of people without whom this project's work would not have been feasible. Their high academic standards and personal integrity provided me with continuous guidance and support. We owe a debt of sincere gratitude, deep sense of reverence and respect to our guide and mentors **Prof. Preeti Shukla** Associate Professor, AITR, for their motivation, sagacious guidance, constant encouragement, vigilant supervision and valuable critical appreciation throughout this project work, which helped us to successfully complete the project on time. We express profound gratitude and heartfelt thanks to Dr Kamal Kumar Sethi, HOD CSE, AITR Indore for his support, suggestion and inspiration for carrying out this project. I am very much thankful to other faculty and staff members of CSE Dept, AITR Indore for providing me all support, help and advice during the project. We would be failing in our duty if we do not acknowledge the support and guidance received from Dr S C Sharma, Director, AITR, Indore whenever needed. We take the opportunity to convey my regards to the management of Acropolis Institute, Indore for extending academic and administrative support and providing me with all necessary facilities for the project to achieve our objectives. We are grateful to our parents and family members who have always loved and supported us unconditionally. To all of them, we want to say, “Thank you”, for being the best family that one could ever have and without whom none of this would have been possible.

Aman Patel (0827CS213D03) Anurag Yadav(0827CS201039) Bhomik Gahlot (0827CS213D08) Chandransh Singh Chouhan (0827CS201062)

Executive Summary

This project is submitted to Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal(MP), India for partial fulfillment of Bachelor of Engineering in Computer Science &amp; Engineering branch under the sagacious guidance and vigilant supervision of Prof. Priyanka Jangde and Prof. Narendra Pal Singh Rathore. The project is a website. In the project, Bing Maps API is used , which is a dynamic map. This project is only for disables. Disables can find the accessible features of property where he/she wants to visit. Also, they can find the route to visit the place and property owners can register themselves and show their accessibility features.

Abstract

The aim of this project was to evaluate and investigate the feasibility of using Microsoft Azure Machine Learning Studio as an alternative for common problem solving strategies today. To this end, a data set was provided. From this dataset models were build in Microsoft Azure Machine Learning Studio, to examine the machine learning algorithms available on the platform's toolbox. After a preliminary study of the available algorithms and data review, it became apparent that the problem fell under the non-linear regression category. Thus, the study could focus on three feasible algorithms; neural network, boosted decision tree and decision forest regression. The major discovery is that the machine learning approach should be suitable for these types of problems due to many aspects. Microsoft Azure Machine Learning Studio is also a good choice for a first step, not the least because of the easily grasped user interface, as well as the wide availability of algorithms within machine learning. Although it lacks features for the more advanced users and there are better alternatives available when designing models with a high performance criteria.

Table of Contents

List of Figures

1 Introduction

Background

Process model development and optimization is one of a the prioritized fields of research for companies in the automation industry. One particular application of interest to ABB Corporate Research Center is automation in maritime industry. Process model development consists of both designing the model and validation. Validation is done by comparing the data obtained by the model with those obtained by testing the systems in real operation conditions. One of the projects at ABB Corporate Research Center within the maritime industry, concerns the development of an propulsion system. Its validation is based on comparing the main characteristics obtained by a numerical model based on first principles and results of tests. Results of this validation will indicate the accuracy of the numerical model. The numerical model for this project is designed under consideration of the system's physics and then fitted with human-designed parameters to the data. Throughout the text this first principle numerical model will be referred to as the primary model. Another important aspect of the project is the development of the model based on data only. The idea here is to create a data-based model and introduce an extra feature of adjusting it to the new data. Such a data-based model could be created using the machine learning approach. An initial investigation was done by ABB using MATLAB toolboxes. As a continuation, it was decided to investigate Microsoft Azure Machine Learning Studio, [1].

Goal

The primary goal of this project is to create and test the data-based model using Machine Learning Studio provided by Microsoft Azure. These models should be tested on several data sets. The second goal is to consider the possibility of introducing self-learning models, improving their accuracy based on new data. To reach the goals of the project, it is required to address the following questions:

- Is it possible to build data-based models in Machine Learning Studio?
- How to build such models?
- Which are the suitable methods?
- Is it possible to extend the concept to have a self-learning model adjusting automatically to produce a better model as extra data becomes available?

Setup

The system has three given input signals and three identified output signals. The task is to design a data based model that predicts these outputs depending on the inputs as shown in figure 1.1.

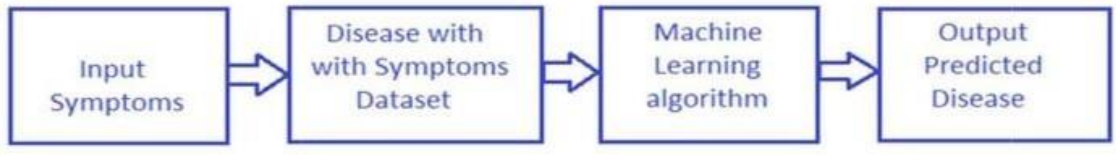

Input Symptoms
Disease with with Symptoms Dataset
Machine Learning algorithm
Output Predicted Disease

Figure 1.1: A schematic of a model with three input and three output signals.

To design the model through machine learning it is necessary to have a set of training data. The machine learning algorithm will use this data set to train the model. [2] The training data is experimental data provided by ABB Corporate Research Center. When a model has been trained it has to be validated by using some auxiliary set of data, called the validation data set. This validation set is also experimental data provided by ABB Corporate Research Center and is in this case a set of data that zooms in on a segment deemed extra relevant by ABB. By comparing the validation data with data obtained by the model, the accuracy can be measured. The auxiliary data set can then also be used as Training data to further improve the model.

The system that is being modeled is a system ABB has created. The input parameters are in this report referred to as alpha, beta and gamma. The measured outputs are delta, epsilon and zeta. It is also relevant for ABB to have a reverse model that can explain what input parameters one should have for a desired output. This reverse model will have the four inputs: delta, epsilon, zeta and gamma, and the two outputs: alpha and beta. Each experimental data point is obtained by specifying the input values and measure the output values in real operation simulation, which is slow and exhaustive. From this produced experimental data the model can be designed. Thus, it is a slow process to produce data points experimentally. As a result, ABB has a limited amount of data. A benefit of having a data-based model is, that it could be able to improve itself as more data become available later on. If the propulsion system is released in the market, customers can use the system and sensors could collect data continuously to gain an even deeper insight. This flow of data could be collected and used to both validate and train a machine learning model. To explore this idea, Microsoft Azure Machine Learning Studio will be used.

2 Theoretical Background

Machine Learning Basics

Given definitions show the difference between data mining and machine learning, two fields that are built upon each other. Many basics can therefore be considered being data mining techniques, while more advanced topics can be categorized as machine learning techniques. Thus, machine learning and data mining are methods to iteratively detect patterns in data, which is not necessarily structured. This also mean that models are not explicitly programmed with a known result. Examples where machine learning is used in big data are diverse. This includes:

- Fraud detection to find anomalies in tax or credit card data.
- Prediction of so-called "rest of useful lifetime" in industrial machines.
- Text sentiment analysis and opinion mining in social networks such as twitter.
- Financial modeling.

Data Mining Workflow

The most basic workflow for data mining, and therefore machine learning, can be divided into six steps. In the first step data acquisition has to be mentioned, as insufficient or biased data can lead to wrong results. In machine learning or data mining this data usually has to be quite big, as patterns might only emerge with thousands or millions of individual data points.

The second and maybe most important step is the cleaning of given data. Problems with data can include unsuitable features, noisy and so on. Features can be nominal (such as yes/no or male/female), ordinal (ranking such as school grades) or numerical (temperatures, cost and other parameters) and sometimes features have to be converted. An example would be to label all days with average temperatures over 10°C as "warm" and all below as "cold". Outliers might either be interesting, as in anomaly detection, or can change the outcome of the learning process negatively, e.g. when using experimental data where outliers have to be removed, therefore this has to be considered in the cleaning process. After all, a scientific or an analytical process can only be as good as the data provided is, as what often jokingly is said: "garbage in causes garbage out".

In the third and fourth step, with the cleaned data and a decision about the chosen modeling technique, is to build and train a model. This is possibly the most abstract step in data mining, especially when pre-built programs such as Azure Machine Learning Studio or scikit-learn (in Python) are used. Simply put, the training process is finding structures in the given training data. What kind of data or features these are is heavily dependent on the goal (clustering or prediction etc.). Taking a simple example of model training: a very famous dataset is the *iris*-dataset, that includes features of three different *iris*-species and uses clustering to determine the correct subspecies. Features included in this set are different parts of the flower, such as petal or sepal length. When using clustering, the training algorithm iteratively calculates the most common features and allows therefore a grouping process of all the data points. The result is heavily dependent on the complexity of these clusters, as is in regression the result dependent on the complexity of a curve. As can be seen in Figure 2.1 it can be quite hard to determine if the bottom-most red data point belongs to the species-A-cluster or to the species-B-cluster, depending on the method of measurement.

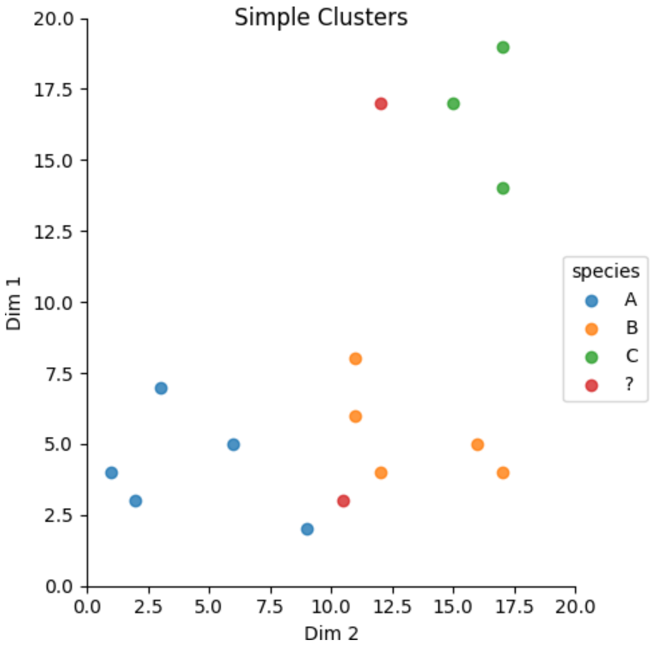


Figure 2.1: A simple, made-up, data set of three different species.

Testing and evaluating the model is mostly done by statistical methods and will seldom give a result of 100 % match between the training and validation data. Considering one of the most intuitive and simple data mining models, linear regression, this uncertainty is mostly covered by introducing a measurement of uncertainty. A good introduction into machine learning and its workflows can be found in [3].

Supervised and Unsupervised Learning

Supervised learning: "Supervised learning is a type of machine learning algorithm that uses a known dataset (called the training dataset) to make predictions. The training dataset includes input data and response values.", [4].

Unsupervised learning: "Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses.", [5].

In data mining and machine learning an abundance of models and algorithms can be found, but most fundamentally these are divided into supervised and unsupervised learning. One fundamental example has been mentioned in the foregoing section, the clustering of *iris*-species. Former is a supervised process where data points are labeled ("species A", "species B" or "species C") and labels are calculated for new data points. Comparing calculated labels according to the trained model with the original label gives the model's accuracy, hence supervised.

Unsupervised learning on the other hand does not require any labeling, since the algorithm is searching for a pattern in the data. This might be useful when categorizing customers into different groups without a priori knowledge of which groups they belong to.

Machine Learning Algorithm Types

For machine learning many different algorithms can be found. A wide variety of these are available in Azure Machine Learning Studio. For simplicity these can be subdivided into four categories, where each category is good for different kind of problems. **Anomaly detection** algorithms, are good for finding unusual data points. Trained **classification** algorithms can be used to categorize unseen data. As an example, it could be used to take in data from a phone on movement to categorize what activity is being performed. **Clustering** algorithms group data into clusters and look for the greatest similarities. This can be used to find unknown connections on huge sets of data. **Regression** algorithms are used to find patterns and build models to predict numerical values from datasets. These will take multiple inputs and determine how much each input affects the output.

Within the regression category there are eight different basic algorithms, each suited for different kind of problems, available in Azure Machine Learning Studio. **Boosted decision tree regression**, which is based on decision trees, where each tree depends on prior trees, uses decision splitting to create stepped functions. It learns by fitting the residuals of preceding trees to improve accuracy. **Decision forest regression** consists of decision trees in regression and is resilient against noise, due to the fact that many trees form a "forest". This makes it easily to parallelize. **Fast forest quantile regression** is effective in predicting weak relationships between variables. Unlike linear regression quantile algorithms try to find patterns in the distribution of the predicted values rather than just predict values. **Linear Regression** is the most classic type which solves linear relationships between inputs and outputs. **Neural network regression** is most common in deep learning and adaptable to regression problems, but might be too complex for simple regression problems and requires thorough training. This method is very stable and is often used when other algorithms can not find a solution. **Ordinal regression** is found useful for predicting discrete ranking. **Poisson regression** is useful to predict values if the response variable follow a Poisson distribution.

In general it should be noted, that the algorithms from Microsoft are not open source and it is therefore impossible to completely comprehend the underlying mechanisms.

Bias-Variance-Trade-off

A problem in machine learning is to find a balanced compromise between training accuracy and validation accuracy. This means to find a function that solves a training problem accurately, e.g. a high-degree polynomial that fits the training data well, but is not overfitting the validation data. On the other hand, a function too simple can oversimplify (underfit) a problem and neglect present patterns. This is a problem evident only in supervised learning and therefore relevant for regression analysis. The dilemma often leads to trial-and-error strategies of finding suitable models [3].

Neural Networks

In the most basic definition, a neural network is a simulation of an animal (human) nervous system based on artificial neurons. Artificial neurons are supposed to imitate the function of a biological neuron; one or more inputs are being summed up and compared to a "firing" threshold that generates an output. The term "to fire" is used in biology and refers to a biological model of neural activity.

Neural networks excel at complex tasks such as language recognition, computer translation or image recognition and what makes their use even more interesting, they are considered to be simply understood and constructed, as the fundamental mathematics behind it is relatively simple. By using the most simple neural network as an example, this section will introduce the basic mathematical concepts behind neural networks. An artificial neuron is, in a simplified sense, a series of inputs. These inputs are weighted and then the products of inputs and weights are summed. Then a threshold comparison decides the output. A single neuron with several inputs and one output is called a "perceptron".

In the following example, a made up data set of berry characteristics shall be used to teach a neural network the distinction between lingonberries and bilberries - the European variant of the blueberry. Characteristics that are measured are color, diameter, and weight. For any human - that has been taught to recognize the two berries - this would be a very simple task, since the colors are so clearly different. Nonetheless, an application might be a factory setup with a robot that has to differentiate tons of berries for packaging. Since color recognition is not an option, the computer has to differentiate between the species by using weight and diameter, that are assumed to be measured quickly and easily.

For a berry, x_1 and x_2 are the inputs to the perceptron. The classification depends on whether $\sum w_i x_i$ is above or below a threshold value. It can be assumed that if the expression is bigger than threshold it is a lingonberry (or '1'), otherwise it is a bilberry (or '0'). To be able to make this differentiation the neural network has to be taught the weights to find as many correct answers as possible. To teach a neural network a training set is used. In the given example 20 berries have been sorted by a biologist and used for training. Starting with random values for the two weights, w_1 and w_2, the learning algorithm will now compare the inputs and (known) outputs of the training set. To find the highest possible accuracy the weights have to be adjusted step by step. As soon as a maximum of berries are correctly recognized, the model can be seen ready. For validation another set can be used to check the accuracy on an even bigger scale. Most of the times it can be seen as beneficial to have very big training sets or to extract outliers and other irregularities. This makes learning in neural networks, or machine learning in general, more difficult, as the data has to be both of high quality and of large quantities.

Neural networks can be used for regression, although, depending on the complexity of the output, it might be excessive. Especially, if the mathematical output function is simple, linear or multiple linear regression might be a better choice, although these tend to easily overshoot functions. Therefore, a different technique is described in the following section. Further information about Neural Networks read [6] and [3].

Decision Tree Regression

Decision Tree is usually a preferred technique to find rules in classification problems. To recycle the berry example from the preceding section, the same assumptions are done to differentiate between the two types of berries. The inputs/outputs in this case are the same, weight and diameter are fed into the system and the output is either a lingonberry or a bilberry. A biologist might, in this simple case, also set up a decision tree, albeit a very simple one: color? If "red" → lingonberry, if "blue" → blueberry. Now, assuming the machine again is provided with just weights and diameters, the tree can grow a little bit more complex, setting first a decision threshold for the weight and then one for the diameter (or vice versa). Which thresholds are chosen, and in which order, is decided by the learning algorithm according to the most efficient information gain. These trees can grow very complex and can make solutions overly complicate, especially in classification.

Decision trees can be used for regression as well. It works as a stepwise function. In the most simple case, let x be increasing discrete values from 0 to 20, and let y be a linear function depending on x. Now, the algorithm might find a step function with threshold $x_{th} = 10$ and set the y value for all values below the threshold to the mean of all y values in the interval $[0 \leq x \leq 10]$, or respectively $[0 < x \leq 20]$. This would generate a simple step with a jump at $x = 10$ from the mean of the first interval to the mean of the second interval, [7]. Very accurate and complicated decision trees, with step functions on many levels, can be found by teaching the model thousands of input/output pairs.

3 Methodology

Data Review

The training data consists of 15 data points, and the validation set consists of 10 data points, both produced experimentally from testing. For each experimental point, rpm, eccentricity and forward speed is being held constant, and the output torque, thrust X and thrust Y are measured. Between the tests it is important to let the test environment stabilize to minimize noise and unwarranted external factors. This waiting take several hours due to the size of the tank and thus these 25 data points translate to several days of testing. Before creating a machine learning model the usefulness of the data must be evaluated. To do this the following four question should be answered, [8]. Is the data:

- Relevant? Can we assume that the input data cause any impact to the output? In this case it is safe to make this assumption. That the eccentricity of the system, the rotation speed and the forward speed all affect the torque and thrusts is physically intuitive. An example of irrelevant data for the machine learning model would be time stamp of each experiment since the condition were made identical between each run.

- Connected? Does each point consist of all three inputs and all three corresponding outputs, or do we have holes? In this case all the data points are fully connected. Disconnected data would be for example that we lacked the eccentricity value for some thrust X, thrust Y and torque. This could be the case if a sensor would break down during long time data collection.

- Accurate? It is harder to evaluate the accuracy of the data since no measurement error is provided. The input data is assumed accurate since they are chosen and display little deviation. By visualizing the output data it can be seen that the torque and thrust X follow some sort of pattern which could indicate that the data is accurate. However, the thrust Y displays no such clear pattern, which could indicate inaccurate data. Furthermore, by comparing data from the training and validation sets for thrust Y you can see that the same input case produces two completely different values for thrust Y which indicate large noise. Thus the output data for thrust Y should not be considered accurate.

- Size of data? Do we have enough data points to train a reliable and accurate data based model? This depends on the complexity of the system's behavior, if we have a perfectly linear single-input-single-output case, fewer data points could

> be enough. In this case we deal with a non-linear multiple-input-multiple-output system. Similar cases usually require several thousands data points to converge to a reliable model. Since the training data set only consists of 15 points it is safe to assume that this will be insufficient to properly train the model.

The lack of data points makes it difficult, if not impossible, to explore the machine learning approach to create a data based model. To circumvent this problem, additional data was requested from the primary model that ABB had designed. This data should not necessarily be considered a perfect substitution of real experimental data since ABB's primary model is also made with these few experimental points as basis. However, with this analytical data we could conduct several tests to indicate how the machine learning approach would perform when more experimental data becomes available. A problem with the analytical data is that the values follow some mathematical model perfectly, such ideal data is not realistic to expect from real operation data. Thus we did tests both on the analytical data as is and with data with added noise.

Software Introduction

Machine learning can be done on several different web services and computer applications. ABB wanted Microsoft Azure Machine Learning Studio to be examined. It is available for free but with limitations on available computational power. To set up, an account has to be created and a login used. There are several tutorials available at the start but the basic structure is that of a big workspace to drag and drop different modules and connect them in the order commands or computations are to be executed. On the left-hand, side categories for different modules can be found, such as data management or machine learning. To start, a new project has to be created, which opens an empty workspace.

Input of data can be done in many ways, either by uploading a data file directly or the use of different modules to get data from the web via APIs. This can even be done automatically with modules. CSV files are the recommended data type but there are modules to convert data to usable formats as long as the character encoding is UTF-8. The data is stored in the Azure cloud but it can be exported to a local computer or dedicated server. If the data is uploaded then the datafile can be used as a module by just dragging it to the workbench and connect it to the modules where the data is to be used. In the same way data generated can be exported manually by saving it to a file on the Azure cloud or downloaded to a local computer. It can also be exported automatically to the cloud or somewhere else via modules for exporting via APIs.

Practical example: Having a CSV file with two columns and the goal to find outliers; first step is to upload a file. Then open a new project. Drag the imported data as a module onto the workspace. Then choose a module under the clustering or anomaly detection category of the machine learning modules. Drag these onto the workspace. Connect them to a module for models, connect the dataset to the model module and press go. Now this is fairly simple, since you might want to evaluate your model as well, this can also be done by selecting a few modules and add to your workspace. The data can then be downloaded or analyzed directly in Azure.

Reference Method

To evaluate how suitable the machine learning approach is, it is helpful to have another approach to compare with. Thus, it was decided to do an analytical parameter model via MATLAB. This model can be found in Section 4 and the underlying script can be found in Appendix A. By having this reference method, discussion concerning the relative performance and suitability of different models can be done. An analytical parameter model was chosen as reference method since this approach is the usual method used when creating models. It was also the method used by ABB prior to this project to create their primary model. In the case of the primary model, it was created by doing a pre-study to determine the first principles of the model, then fit it to the available experimental data. These two methods are fundamentally different and most machine learning models can be considered and nonparametric.

The analytical parameter model requires human ground work, where the model designer usually first visualizes the data and considers the science behind the system to create a parametric model template, without specified parameters. Some popular examples would be a polynomial model, an exponential model or a Fourier series. Then that model template is being fitted to a set of data by minimizing error by some definition, commonly least square error. The choice of parameter model template and error definition causes a human bias in how the final model might turn out based on the experiences and preferences of the designer. A way to minimize such bias would be to do a cross-over analysis to determine the most accurate model. However, this usually requires a lot more work and there would still be a human bias since different designer may have different preferences when it comes to performance evaluation. Since two different designer always could design two different analytical parametric models for the same problem, there is no one analytical model that alone can be designed to represent the entirety of the analytical model approach. However, by comparing the machine learning approach with at least one of these infinite number of potential analytical models, there could still be some important insights to gain.

Reverse Model

Another goal of this project was to examine how to set up a reverse model of the system. By reverse model it is implied that the output parameters of the system is set as the input of the model while the input parameters of the system is set to the output of the model. Specifically for this project, the parameters delta, epsilon, zeta and gamma are the inputs of the reversed model, while the alpha and beta are the outputs.

The purpose of having this reverse model is to provide information for the control of the system. If there is a specified output that is desired it is useful to have a model that predicts what input should be provided to achieve this. A reliable reverse model could be used instead of or as complement to automatic feedback control and has the advantage of not requiring active output sensors.

Principally, there is no difference in setting up the design of a reverse model in Azure Machine Learning Studio. One simply switches the output and the input in the training of the model. The only difference is that a system usually only has one output for each input, however a system may have several inputs that cause the same output. This ambiguousness is impossible for any reverse model to predict. This was not the case in this system since it was checked that each combination of output had an unique combination of input. Hence, the reverse model was designed in a similar way as the ordinary model.

The simplicity of reversing an Azure Machine Learning Studio model is an advantage it has compared to analytical models. To design a reverse analytical model it is not as easy as to change input and output and do a refit of the parameters. Usually, analytical models have some mathematical functions included that has to be reversed manually. For example, an exponential function need to be changed to a logarithmic function. This process of manually reversing analytical models can for complex models be very difficult, maybe even impossible. This process is skipped altogether in machine learning models since they do not consider any predefined functions when trained.

4 Results

Results in Machine Learning Studio

One result is the setup of a model and training in Azure followed by testing how the model works for different sets of data points and different amounts of data. As well as testing the speed of the different models and the accuracy of the results each model produce. The first image shows the setup in Microsoft Azure Machine Learning Studio and the following graphs the results are shown. These will be discussed in the discussion section.

Our underlying data was generated by a deterministic algorithms without noise addition. This will create very smoothly distributed data which will give much better results in testing than what real data would accomplish. In order to illustrate this phenomenon there are two different plots: one plot for a method where the data is the given data and another plot where we have introduced a Gaussian distributed noise factor. After illustrating this we also made a plot to show the difference between the models using a much larger dataset.

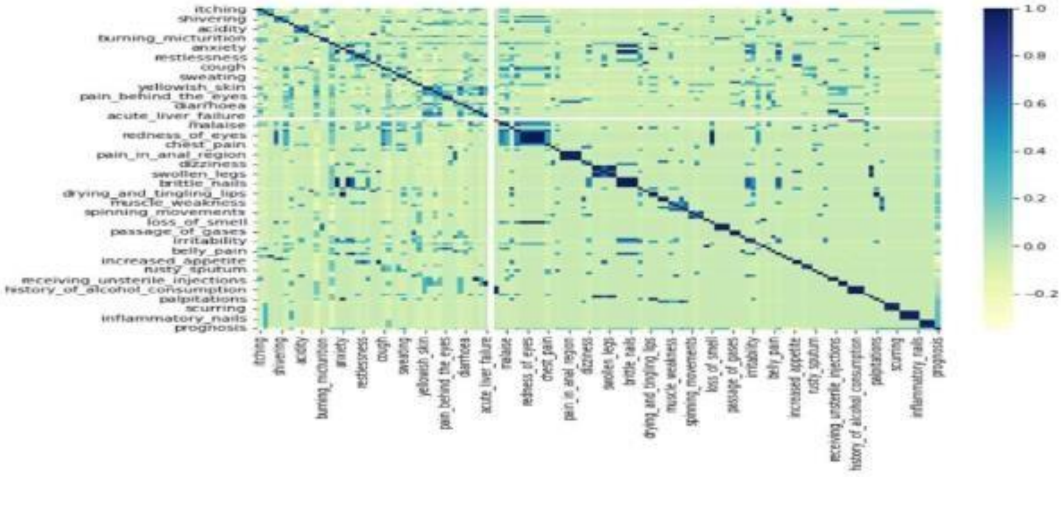


Figure 4.1: Azure ML Setup

In Figure 4.1 the more complex array of different modules in Microsoft Azure Machine Learning Studio can be seen. This model can considered the most interesting result, since it shows how the software runs and compares different algorithms. These algorithm blocks are ordered from the left to the right branch and get fed and trained with the same input data and saved in data structures when processed.

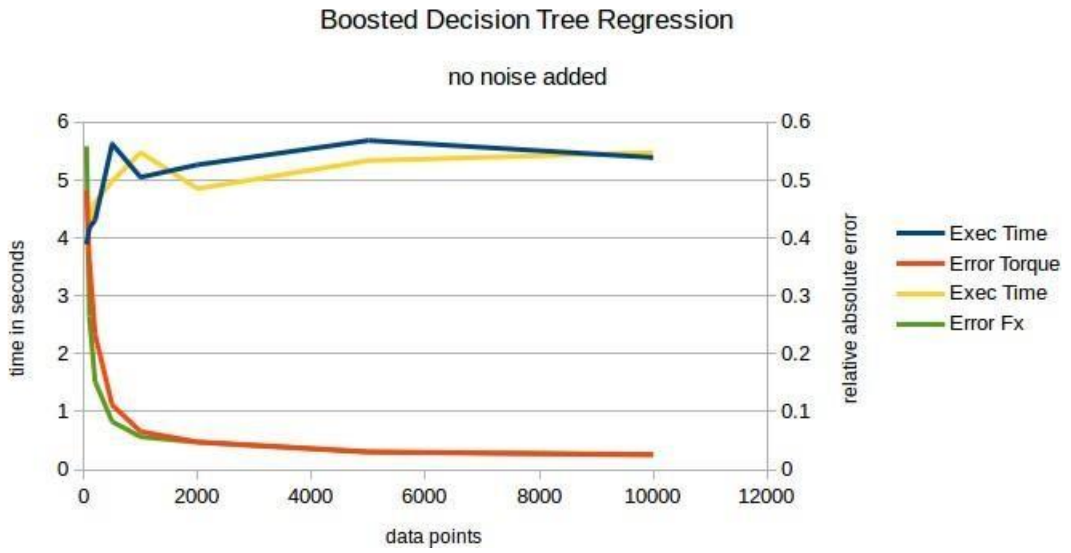


Figure 4.2: Boosted decision Tree without noise

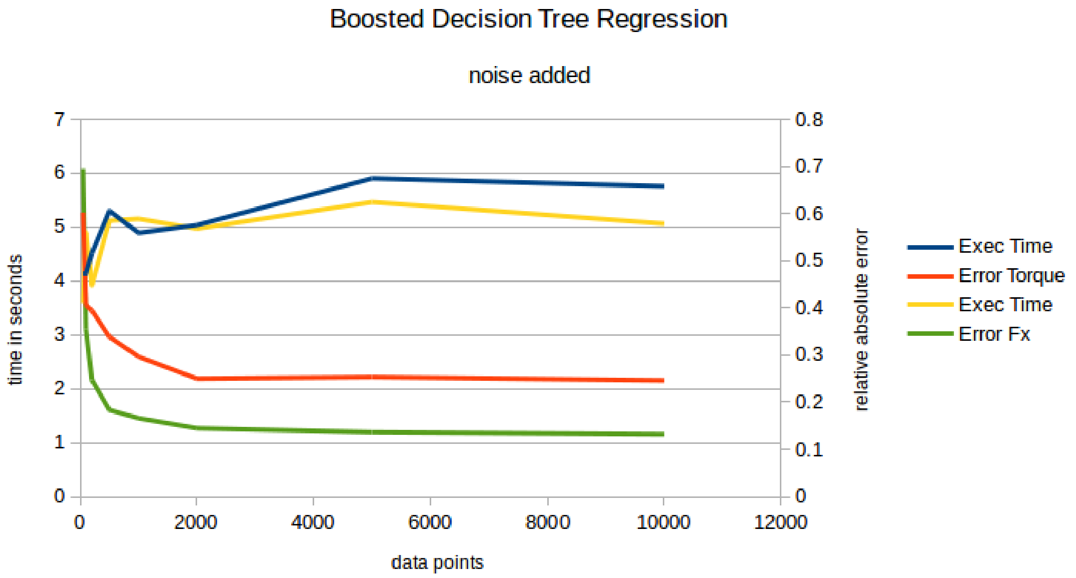


Figure 4.3: Boosted decision Tree with noise

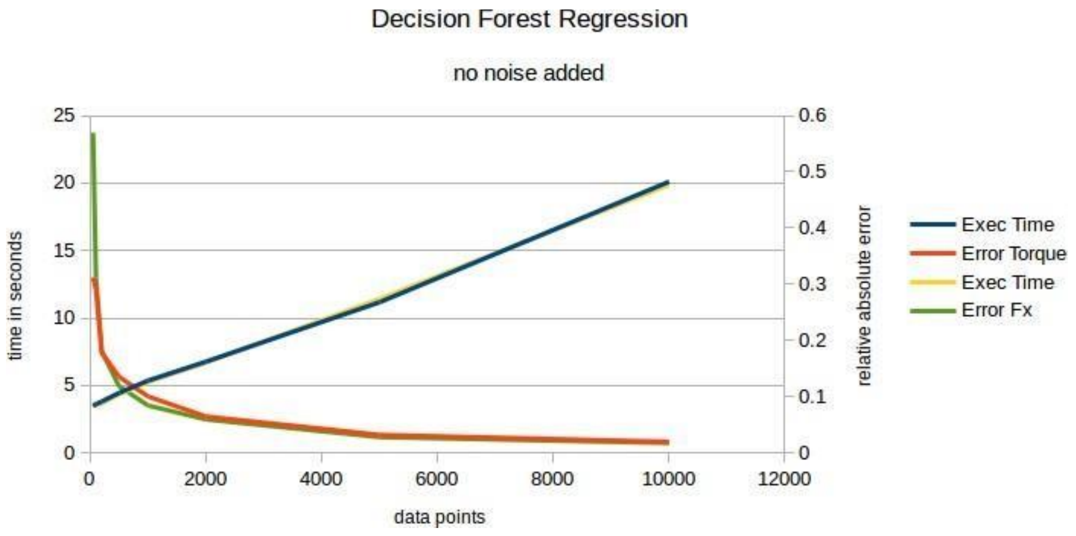


Figure 4.4: Decision Forest Regression without noise

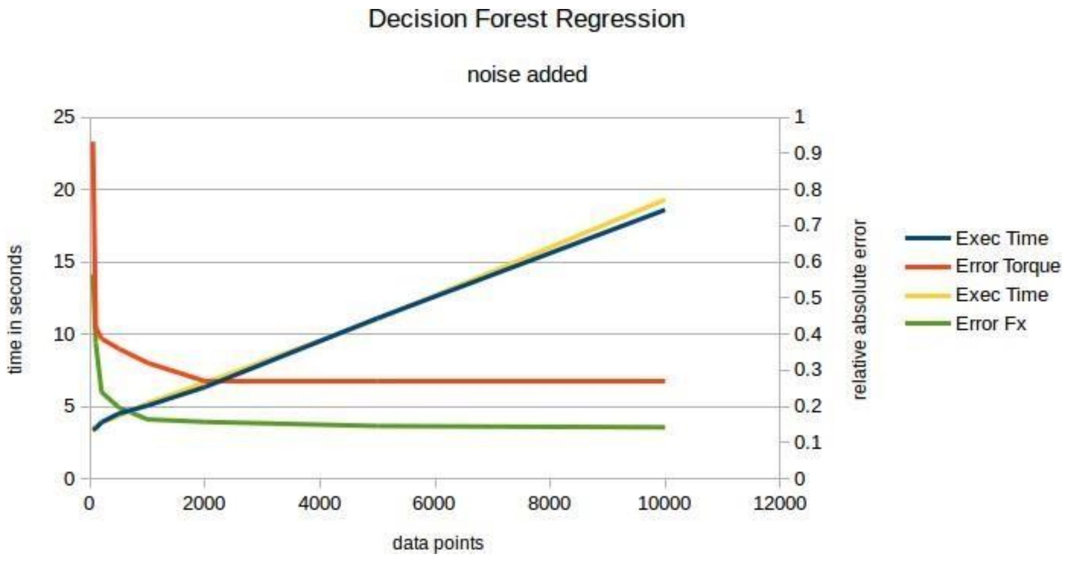


Figure 4.5: Decision Forest Regression with noise

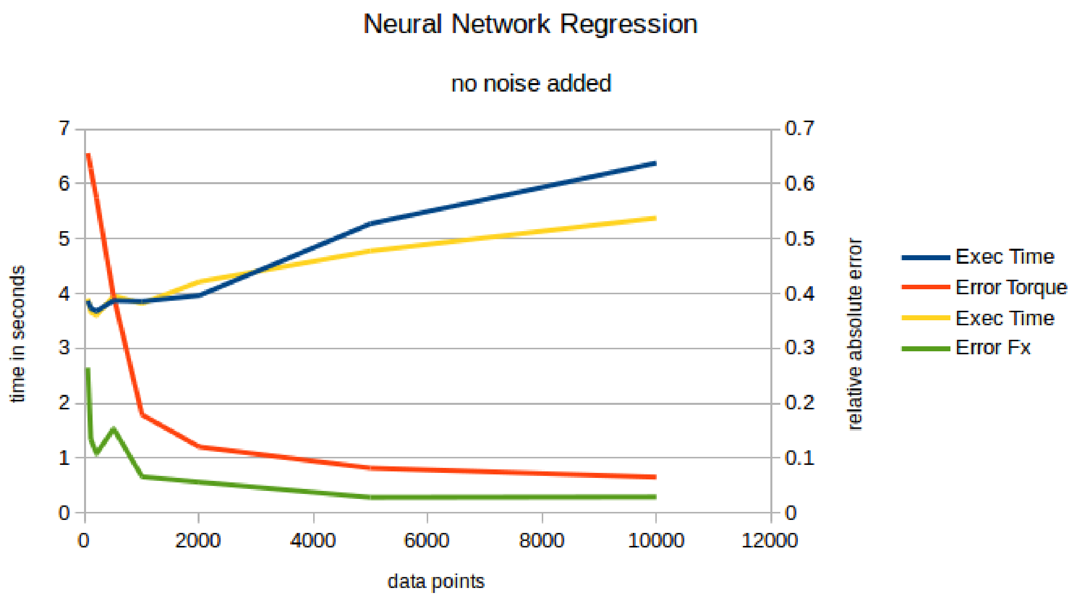


Figure 4.6: Neural Network without noise

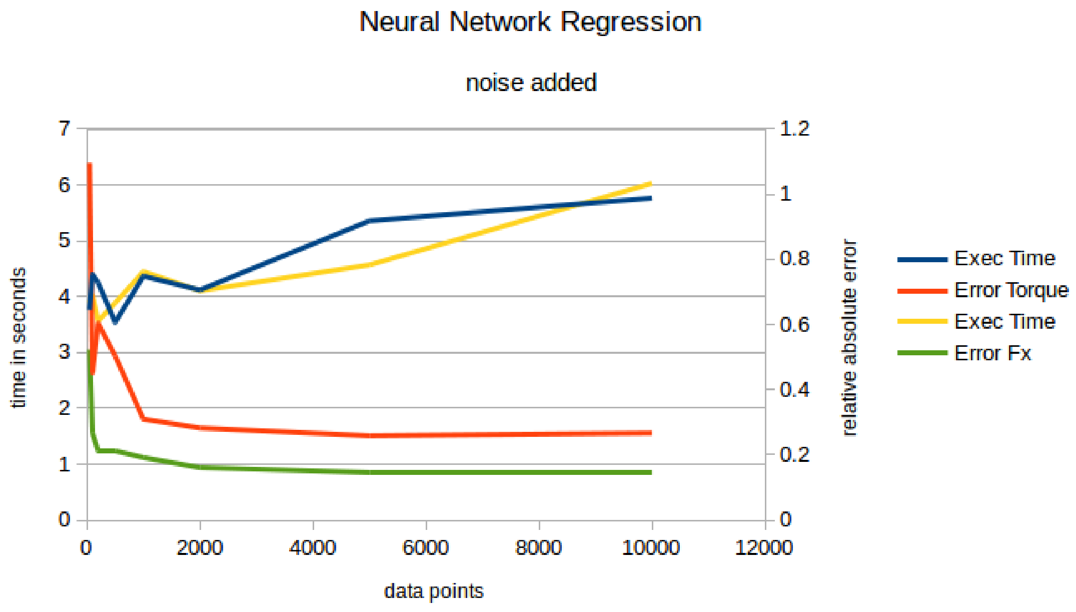


Figure 4.7: Neural Network with noise

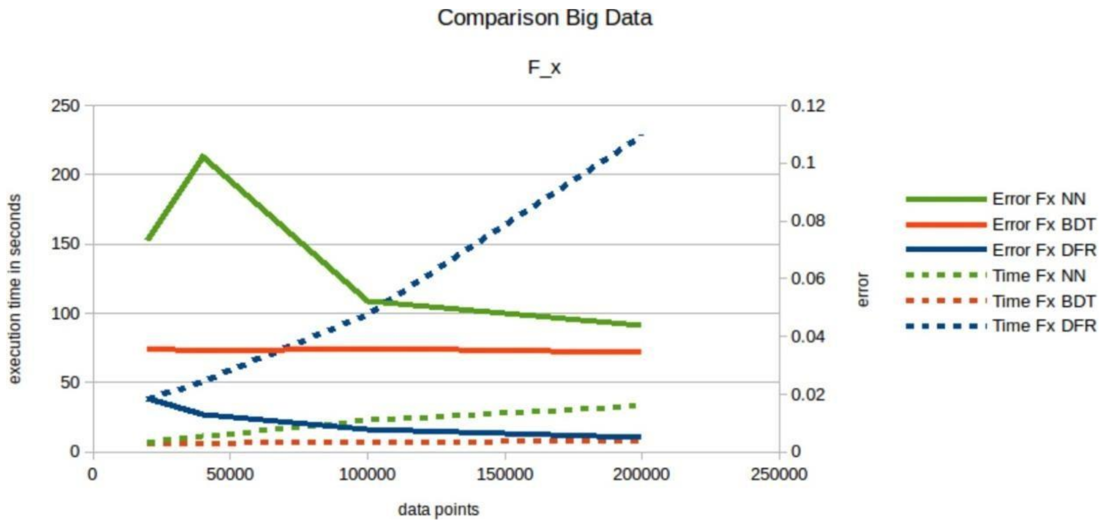


Figure 4.8: Time to build model and accuracy for the different algorithms on large set of data, here to predict F_x values.

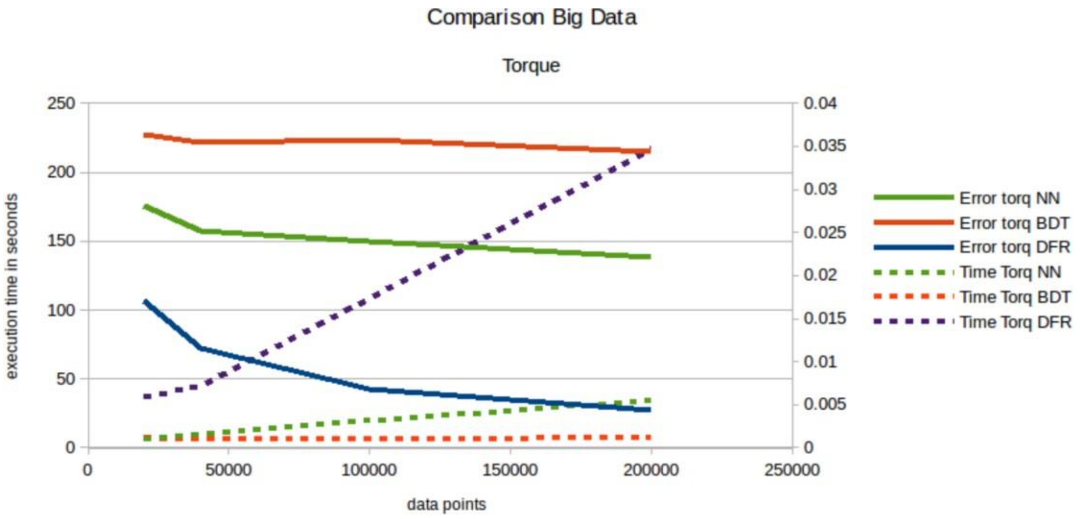


Figure 4.9: Time to build model and accuracy for the different algorithms on large set of data, here to predict Torque values.

Results in MATLAB

Analytical model

In order to properly evaluate the machine learning approach to design a model for this system, it is important to compare it with other approaches of model design. In particular, it would be interesting to see how machine learning models perform compared with analytical models by human design. Since there are infinite number of ways an analytical model can be designed, there is no one correct model that can represent the whole approach of designing an analytical model. However, by designing one analytical model there are still many insights that can be obtained by comparing it with the machine learning models.

The following analytical model was created using MATLAB. To design the model for the multiple-input-multiple-output system several multiple-input-single-output systems were designed and then recombined. The first multiple-input-single-output system has Alpha and Beta as input and Zeta as output. The first step is to plot Zeta of the training data as a function of Alpha for each case of Beta.

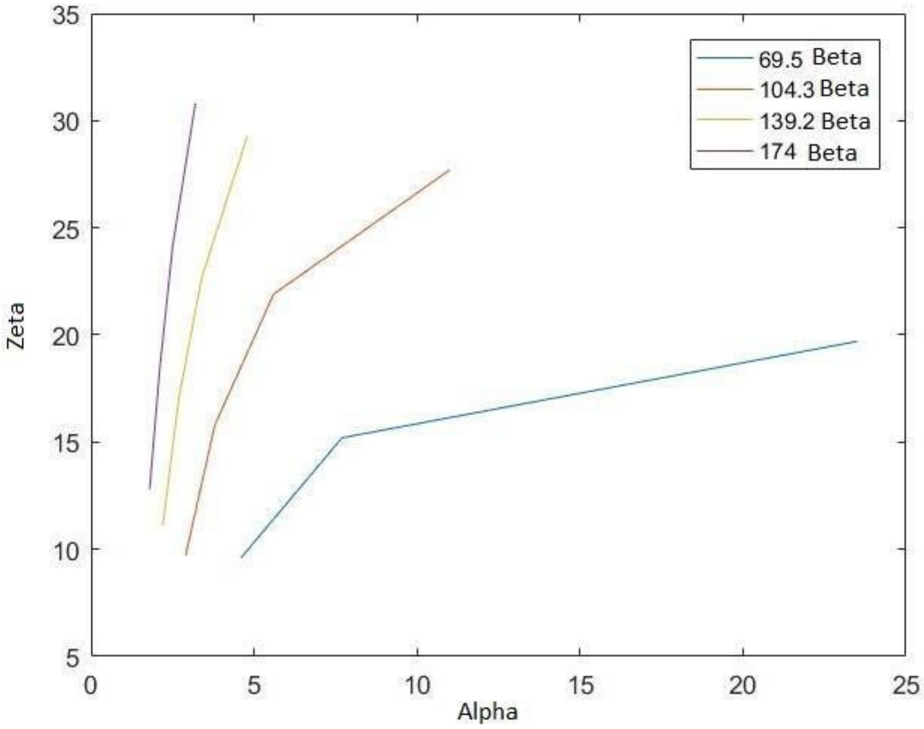


Figure 4.10: How Zeta changes as a function of Alpha for each case of Beta

From Figure 4.10 you can see a pattern. Zeta increases fast for low values of Alpha but seems to stabilize to some asymptote. For such behavior a power fit was chosen for each function.

$$Zeta = a \cdot Alpha^{b} + c \tag{4.1}$$

Each case of Beta produced different values for the parameters a, b and c in equation 4.1. To tell how these parameters depend on Beta each of them was plotted against their corresponding Beta.

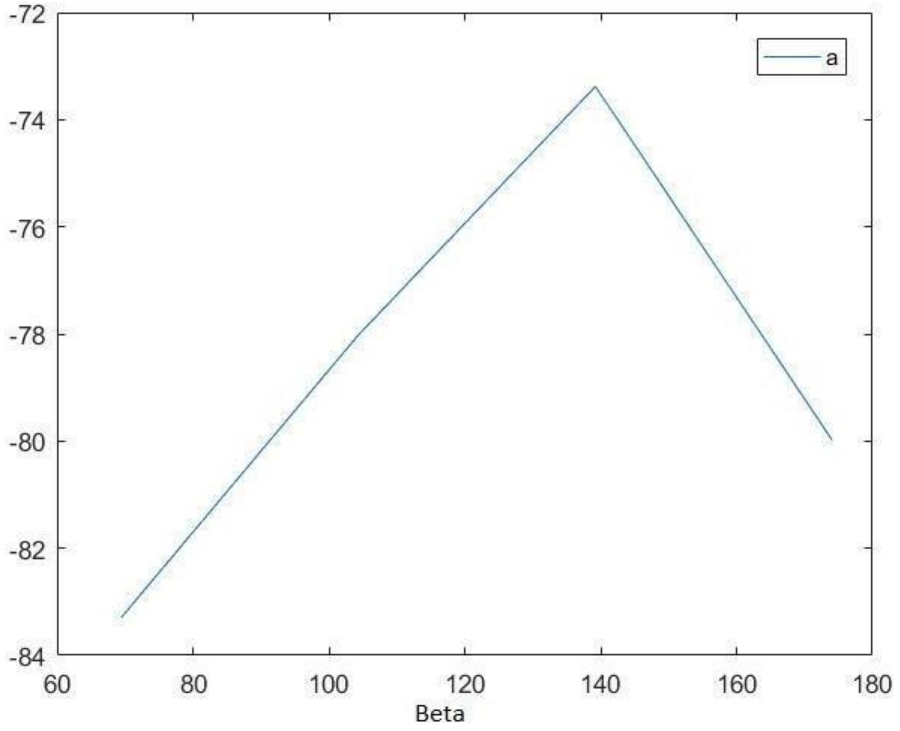


Figure 4.11: How parameter a in equation 4.1 changes with different Beta

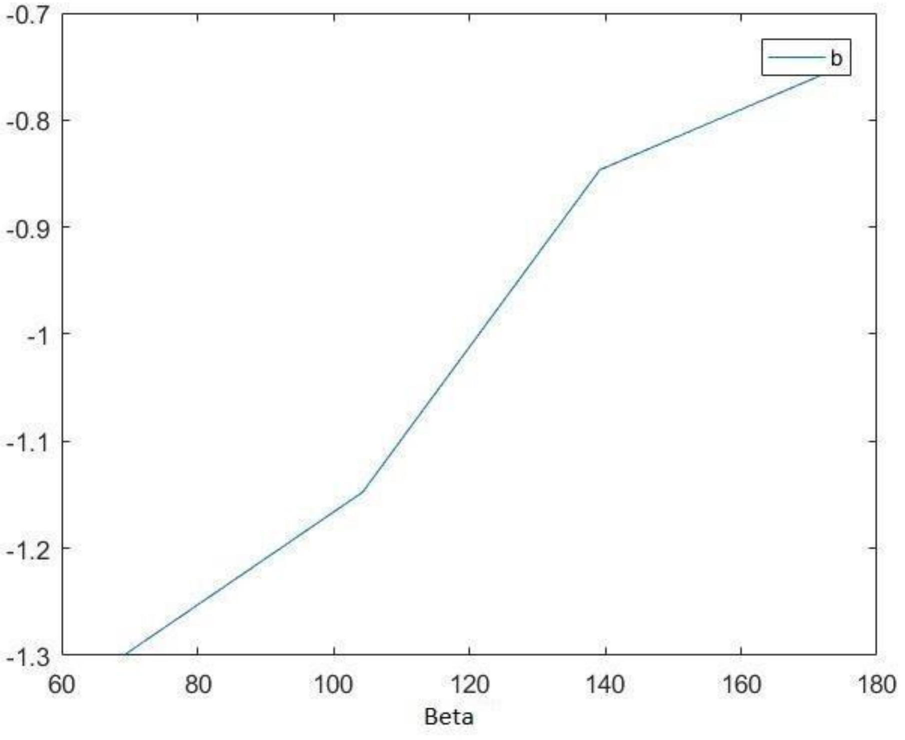


Figure 4.12: How parameter b in equation 4.1 changes with different Beta

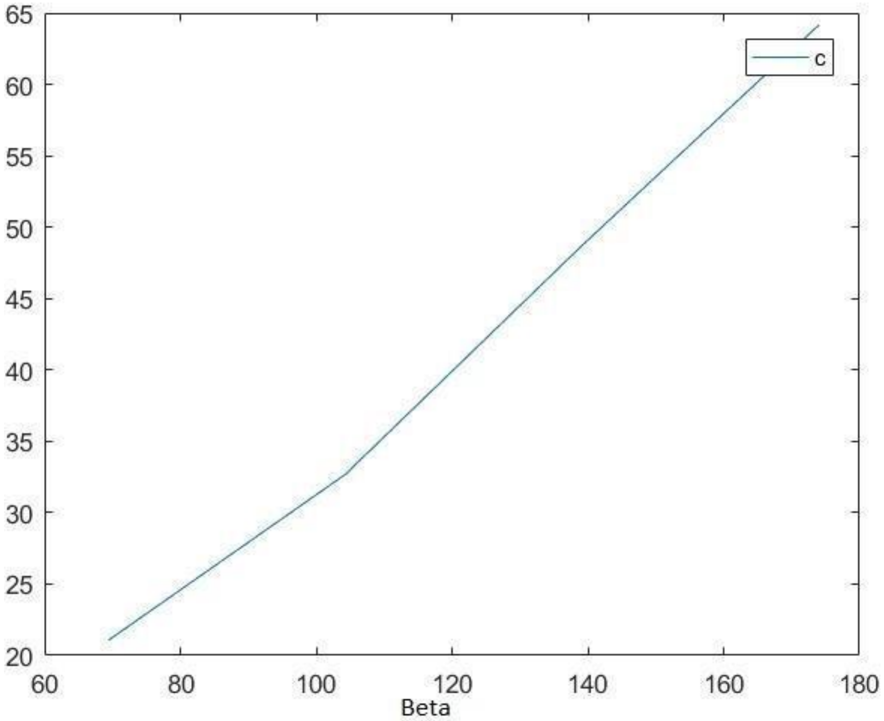


Figure 4.13: How parameter c in equation 4.1 changes with different Beta

From above figures a appears to be a semi constant around its mean value while b and c are approximate linear functions of Beta. Using a linear fit to predict b and c as a function of Beta and taking the mean value of a and put them back into the power fit equation 4.1 we get the initial model.

$$Zeta = a \cdot Alpha^{b_1 \cdot Beta + b_2} + c_1 \cdot Beta + c_2 \tag{4.2}$$

With the parameters $[a, b_1, b_2, c_1, c_2] = [-78.6694, 0.0056, -1.6889, 0.4168, -9.0828]$. To further optimize the parameters, a surface fit is used with equation 4.2 as a template function and the parameters as an initial guess. The final analytical model is then produced.

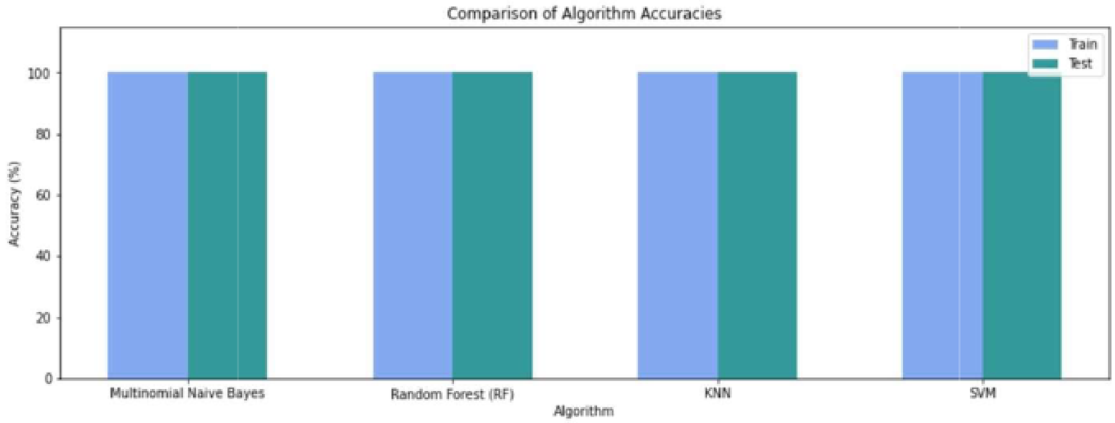


Fig. 12 Accuracies of algorithms

Figure 4.14: The analytical model of Zeta as a function of Alpha and Beta compared to the training data points

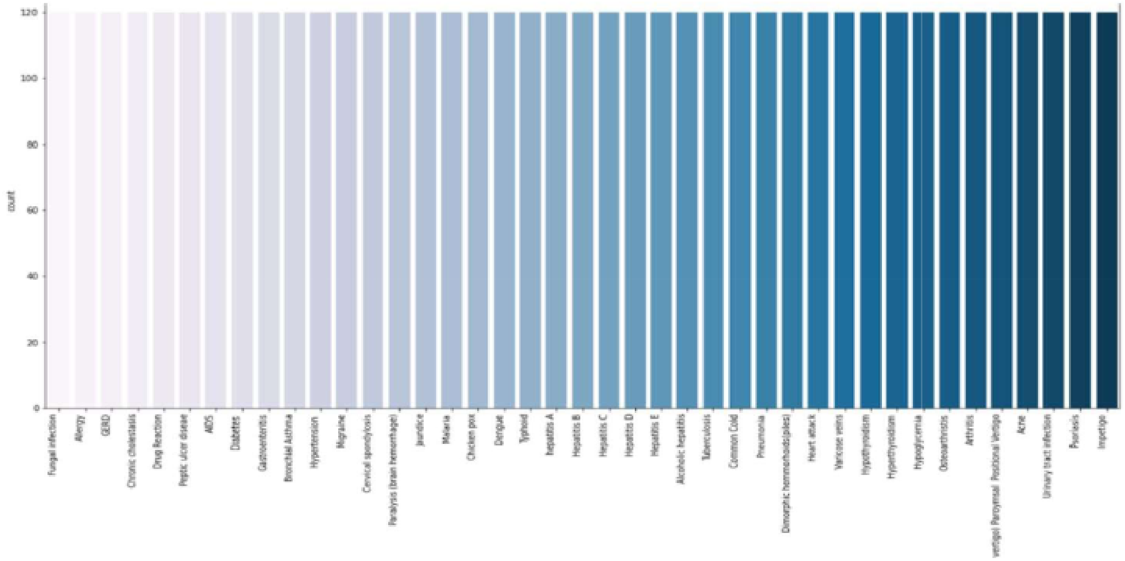


Figure 4.15: The analytical model of Zeta as a function of Alpha and Beta compared to the validation data points

Comparing this analytical model with the validation data you obtain an error of 2.5%. A similar process is repeated to create the multiple-input-single-output system that predicts the Delta. However, for Epsilon the data behaved too irregular to chose a good fit function.

5 Discussion

Microsoft Azure Machine Learning Studio review

Cloud service

While commonly "the cloud" is connected to Dropbox or Google Drive applications, companies such as Amazon (Amazon Web Services), IBM, Google and Microsoft have started offering sophisticated cloud environments. These include infrastructure-, platform- and software-as-services are often abbreviated IaaS, PaaS and SaaS. Within the scope of this project the Azure Machine Learning Studio is to be used primarily, which is a software-as-a-service application. While formerly software licenses were distributed by handing out activation keys, the cloud model allows a pay-as-you-go model, where e.g. used traffic, user access or messages sent will be accounted for. Furthermore, this means there is no need to own physical computing resources, even if big data is to be handled, which reduces costs for maintenance and acquisition. While cloud services offer simple access to hard- and software, a huge challenge for companies is to assure data security over their disclosed data. One thing to note for industrial use is that since Azure is a cloud based platform all data is stored on the cloud. Even though it might be encrypted and safely stored there will be a question of security to investigate.

User experience

To get started with Microsoft Azure Machine Learning System all that is required is an Internet connection, a computer and an account. The account is free of charge but the user get limited to one CPU in the cloud. So with bigger projects it will take time to execute. At first impression the Machine Learning Studio is a bit overwhelming and confusing. There are many tutorials and videos to guide the user to the first project and getting started right away. The only problem here is that there are so many different "getting started" videos that it is hard to know which one to actually start with. Once past that initial barrier the UI is quite intuitive. Since the program is based on modules it is easy to do what you want, once you know exactly what that is. The modules often offer customization which allows the user to tailor the program a bit. But here lies also one of the bigger draw backs of the Machine Learning Studio. Exactly what goes on in the modules is secret, which limits the user adaptability and optimization of the program the user is making. However, it is very good for a user-friendly overview and introduction to what machine learning is and what it can do. Since Machine Learning Studio is so user friendly, the technical know how of the user can be quite low. The more challenging part is to understand what is going on, and design programs for particular purposes.

Comparison of Methods Analytical vs ML Model

In engineering problems one might nowadays consider not only analytical solutions but also use data mining, or automatized machine learning, to create models. Often the engineer will be confronted with regression problems, which can be troublesome in their complexity: overfitting and wrong assumptions can lead to a very inaccurate behavior on unseen data. On the other hand some methods are, by far, too simple to paint the correct picture, so-called underfitting. Machine learning might offer an escape with reservations. In that, “machine learning” does not refer to the more simple regression techniques, such as linear or polynomial regression, but to more crafty techniques, such as neural network regression or decision tree regression.

The biggest argument for such an approach is that the autonomous learning does not require any assumptions about the shape and behavior of the function. Decision trees and neural networks just categorize unseen data points according to rule learning or similarity not according to some assumption about the functional behavior. In analytical modeling on the other hand, assumptions have to be made regarding the complexity of a function, and correct fitting can be a time-consuming task. This can be seen for example in MATLAB’s fitting-curve module. To create an analytical model different approaches had to be tested manually, to determine if an exponential approach fits better than e.g. a third-order polynomial.

The biggest argument against a machine learning approach might be that an engineer has to rely, to some degree, on a black-box approach. Since machine learning can be seen as a learning process from data without explicit programming, the user needs deeper understanding of the algorithm and some trust in to this methods. Of course one might argue that for this validation techniques have been developed, but these are not fool-proof either. Other factors that might play a role in whether a machine learning approach is working or failing, is the quality of both the learning and the validation data. Data quality means in this case not only the accuracy of the measurement but to a high degree also the quantity and the range of the data: many data points over a wide range of input parameters.

Why is machine learning then so "hyped" [9] and successful? The answer to this might lie in the diversity of its applications. Outside of regression machine learning thrives for example in pattern recognition – visual patterns, acoustical patterns, data patterns – and recommendation models. These are applications where great amounts of data ("Big Data") is collected.

Surrogate modeling

Because of the limited amount of experimental data that was available, it was virtually impossible to create an experiment based machine learning model. To be able to explore the machine learning approach, a surrogate model had to be created. By definition, a surrogate model mimics the behavior of an underlying model, [10]. The underlying model can sometimes by called the simulation model, the output model or as in this case, the primary model. A surrogate model is an engineering method used when an outcome of interest cannot be measured easily, so a primary model of the outcome is used instead. In this case, ABB had a primary model of the system. By requesting a large amount of data produced by ABB's primary model, there was a sufficient amount to properly train the machine learning models. Since the models created in this fashion do not attempt to mimic reality, but rather try to mimic the behavior of the primary model, these models become surrogate models. By examining the performance of the surrogate models created by using different machine learning algorithms, insights regarding the algorithms can be reached. However, there are some assumptions and limitations that must be considered when drawing conclusions from surrogate models.

- **Authenticity of the surrogate model** A surrogate model can never be more accurate than the model it tries to imitate. That means that if the primary model is completely false, the surrogate model cannot become true no matter how well it processes the data. Because of this, the ability of the surrogate model to predict the real world, its authenticity cannot be validated unless the authenticity of the primary model has been validated. Since we have no means to examine the authenticity of ABB's primary model, we cannot examine the authenticity of the surrogate Machine Learning models.

- **The convergence of the surrogate model** A main desirability of surrogate models is to be able to mimic the behavior of the primary model as computationally cheap as possible. A primary outcome model can be very complex, rigid and computationally expensive. Lets say a primary model required extensive theoretical research and used millions data points to be designed, if a surrogate model could mimic that model closely by only using a couple of thousands data points, the surrogate model could be preferable to use for future simulations. By comparing the convergence of different machine learning algorithms to the same set of analytical data conclusions on the suitability of each algorithm can be drawn.

- **The surrogate model with non-ideal data** A large problem with the use of a primary model data to train an surrogate model is that the model data becomes ideal. That a surrogate model converges quickly to ideal data does not mean it will perform well with non-ideal data. By 'ideal' it is implied that the data is homogeneous throughout the experimental scope, is perfectly tangent to the model and without stochastic elements. To deal with this problem, artificial impurities can be added to examine the performance response. In this case, white noise was added to the data to see how the models behaved. However, homogeneous white noise throughout the data is a poor imitation of the chaotic nature of reality. In reality, noise will behave differently throughout the experimental scope, furthermore the density of data may also vary, and there could very well exist significant holes in the data where experimentation is extra difficult or less interesting. Thus any behavior the surrogate model displays is only an indicator on how the model could behave on real operation data.

Choice of algorithm

Here we will discuss the results in section 4.1 considering the algorithm theory in chapter 2. First of all it is important to note that performance of the Linear Regression and Gaussian Linear Regression algorithms are not included in the results. This is because those method were clearly unsuitable for predicting the behavior of this system. This was to expect since the system is non-linear, which could be seen by visualizing the data.

The error converges around 3000-4000 data points for each of the algorithms. This could indicate that approximately 4000 data points could be enough to train the machine learning model on this system. There are however some important differences between each algorithm performance. Both Boosted Decision Tree and Decision Forest converges similarly to a notably lower error than neural network. The similarity between their boosted decision tree and decision forest could be due to their similar algorithmic layout. The reason why the Neural Network algorithm does not achieve the same low error could be due to choice of hyper-parameters not being optimal, such as the number of hidden nodes. This would add a human design aspect to neural networks. The execution time of boosted decision trees remain fairly constant with respect to the number of points. This find could be due to fact that you before hand determine the maximum and minimum number of leaves used, if the algorithm used the maximum amount of leaves after 1000 points the computational cost of every case with more data would more or less remain the same. Decision Forest Regression display a clearly linear execution time with respect to the number of data points, which is also to expect since decision forest regression will adapt its complexity to number of data.

The added noise did not seem to slow down the convergence. This might indicate that the model would be reliable enough even for real operation data which might not be ideal. It would be more suitable to have otherwise randomized noise or impurities, e.g. cutting out chunks of data, to see how the algorithms might perform. As we did tests with a large data set to see the algorithms behavior with larger numbers of data,

>10000 points, we noticed the error continuing to go down for boosted decision tree but converging for the other algorithms. We believe this is due to the primary model generating smoothly distributed data, which is fitting in the way boosted decision trees algorithm works. With noisy data this behavior might not reproduce.

6 Conclusion & Future research

Recommendation

The main goal of this project was to determine whether or not machine learning is a suitable method for designing a model of ABB's propulsion system, and by extension similar systems. From both experimentation and research the overall verdict is: yes, probably. If ABB manages to produce more experimental data, machine learning could be a very useful tool to design the model. The advantages and disadvantages of this approach compared to the analytical is detailed in the section 5.2. The reason the verdict is not a certain yes is that this could not be tested on real experimental data, but by creating a surrogate model of ABB's primary model. Because of this the results are only indicators of the machine learning performance, see 5.3. If ABB were to use machine learning in future model design the three algorithms: neural network, decision forest, and boosted decision tree, all of which seem to be valid choices. See 5.4 for a complete comparison. Another goal was to examine Microsoft Azure Machine Learning Studio as a suitable platform to use for machine learning modeling. The verdict here is: probably not. Microsoft Azure Machine Learning Studio could be used as a first step to quickly design a preliminary model and get some idea how you want to proceed when designing the final model. But because of the limited insight and design options it is less suited for creating models of complex systems with the high performance required in the technological industry. To read more about this see section 6.2.

Alternatives to Microsoft Azure Machine Learning

Microsoft Azure offers a quick start with a drag & drop interface, allowing the user to simply create machine learning models from scratch. Furthermore there is a variety of interactive tutorials introducing machine learning and data processing in Azure Machine Learning Studio. While results can be obtained easily in Machine Learning Studio, it does not offer too detailed explanations of what algorithms are used. The latter is also due to the fact that Microsoft is not publishing the algorithms used in Machine Learning Studio. To have a look into offline-methods, one very popular to come to mind, is the open-source solution scikit-learn, a Python implementation for data processing and machine learning.

From personal experience the authors conclude that scikit-learn scores with a great variety of tutorials, open-source algorithms and a steep learning curve, provided enough motivation to work on it. On the other hand it requires more software-tweaking and ran better under Linux, as the whole set-up process is held more simple than in Windows 10. On a closer look scikit-learn actually runs on the same process-flow (pre-processing, model learning, scoring, evaluation) in its code structure. This is due to the high level of the scikit-learn syntax. For better performance scikit-learn might also require familiarity with the pandas package – and more specifically with the DataFrame data-structures. If one is not to run scikit-learn for machine learning, the tutorials can be recommended nonetheless, as they require little understanding of the syntax and are built upon practical examples. From scikit-learn it is just a few steps into deep learning, since Google's TensorFlow package is also developed for Python and bears some similarities with scikit-learn. TensorFlow is specifically developed for deep learning, meaning neural networks running with at least one and more hidden layers.

A quite different approach is taken by the RapidMiner software, a proprietary software, that offers the same drag & drop principle as Machine Learning Studio but renounces the use of a cloud platform. This software has been used by group members for different courses in Data Science and can therefore be only evaluated from a general perspective. While it offers much more selection on algorithms and pre-processing possibilities, it bears also the flaw of being proprietary, since the free edition is limited to 10,000 data points as input, which is often not enough for proper machine learning. It also does not offer, in any version, data-stream handling.

The last comparison made is with Amazon Web Services Machine Learning. The authors want to emphasize that this comparison is only based on personal experience and has not been done thoroughly enough to make a final conclusion on usability. Amazon Web Services offers less possibilities on machine learning and does not come with a GUI as Machine Learning Studio does. Therefore it is not as nice a playground for beginners in machine learning. Amazon Web Services focuses more on highly automated predictive learning rather than a broad availability and therefore lacks for example clustering algorithms.

There are more platforms out there, such as Amazon SageMaker that offers more sophisticated solutions than Amazon Web Service Machine Learning. Two other frameworks originate from Google, one of them based on TensorFlow (Google Machine Learning Engine). For a more detailed comparison see [11].

Future research propositions

- **Test machine learning on real data**: When ABB has produced a couple of thousands experimental data points it would be a good idea to redo this project to determine the suitability of Machine Learning. Alternatively, an another system could be examined, one that already has a lot of data to it.
- **Try out other machine learning platforms**: Because of the current popularity of Machine Learning there are many platforms available. Python, MATLAB and Google's TensorFlow could all be interesting to explore.
- **Set up sophisticated data-driven experimental structures**: to collect clean and useful data is a challenge, even more if the results have to be very accurate. To use big chunks of data, processes have to be set up, to ensure high quality and quantity of data, access to it and the ability to draw conclusions from it. To implement real-time analytics there are software solutions such as Spark, Hadoop or similar software offered by Amazon Web Services and Microsoft Azure.

Literature

[1] K. Mishchennko, “Microsoft Azure Machine Learning for Data Based Modeling”, 2017.

[2] T. M. Mitchell, *Machine Learning*, 1st. McGraw-Hill Science/Engineering/Math, 1997.

[3] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques for Building Intelligent Systems*. 2017.

[4] (2017). ’Supervised learning’, Mathworks, [Online]. Available: https://se.mathworks.com/discovery/supervised-learning.html (accessed January 31, 2018).

[5] (2017). ’Unsupervised learning’, Mathworks, [Online]. Available: https://se.mathworks.com/discovery/unsupervised-learning.html (accessed January 31, 2018).

[6] (2017). ’Overview of artificial neural networks and its applications’, Xenonstack, [Online]. Available: https://www.xenonstack.com/blog/data-science/overview-of-artificial-neural-networks-and-its-applications (accessed January 31, 2018).

[7] (2017). ’Decision tree regression’, scikit-learn, [Online]. Available: http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html (accessed January 31, 2018).

[8] B. Rohrer, “Is your data ready for data science?”, *Data Science for Beginners*,

[9] (2017). ’Top trends in the gartner hype cycle for emerging technologies, 2017’, Gartner, [Online]. Available: https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/ (accessed January 31, 2018).

[10] N. Queipo, “Surrogate-based analysis and optimization”, *Progress in Aerospace Sciences*, 2005.

[11] (2017). ’Online ml comparison’, Altexsoft, [Online]. Available: https://www.altexsoft.com/blog/datascience/comparing-machine-learning- as-a-service-amazon-microsoft-azure-google-cloud-ai/ (accessed January 31, 2018).

A Appendix

SECTION TITLE

Table A.1: Table of measurements for large set of data

Nr of points	20000	40000	100000	200000
Error Torq NN	2.8106E-2	2.5217E-2	2.3973E-2	2.2152E-2
Error Torq BDT	3.6331E-2	3.5512E-2	3.5757E-2	3.4409E-2
Error Torq DFR	1.7091E-2	1.1556E-2	6.8760E-3	4.4219E-3
Time Torq NN	6.4870	9.9019	19.4849	34.0009
Time Torq BDT	6.7859	6.0789	6.069	7.0709
Time Torq DFR	36.8740	44.7539	108.276	216.6440
Error Fx NN	7.3172E-2	0.102122	5.2398E-2	4.3639E-2
Error Fx BDT	3.6006E-2	3.5094E-2	3.5729E-2	3.4594E-2
Error Fx DFR	1.8405E-2	1.2881E-2	7.7299E-3	5.0359E-3
Time Fx NN	6.883	11.035	22.645	32.9470
Time Fx BDT	5.4489	6.0679	6.4749	8.0109
Time Fx DFR	37.747	50.811	99.0049	228.24

Using the full Azure potential in Automatising

Motivation: Assuming that sensors and interconnected systems or long-term experiments create huge amounts of data, processing (and learning) should be automatised as far as possible.

Approach: The approach is to utilise the Microsoft Azure cloud environment, that not only offers infrastructure-as-a-service (networks, servers ...) but also platforms and software for data management and messaging services. Relevant services that have been tested here are IoTHub (IoT platform), Stream Analytics (Messenger), Visual BI (Visualisation), Blob Service (Storage) and Machine Learning Studio.
This only gives a short overview over possiblities besides the often-mentioned Machine Learning Studio and shows how Azure could unfold its full potential.

Concept: The main idea of connected sensors and devices is that instead of saving data points into local files and on local storage media, cloud environments such as Azure can offer everything required for setting up a work flow like this: Get data, message data via a network connection (the so-called IoTHub (Internet of Things) here) and a messaging service into a storage container (blob storage) or to furthermore visualise and process this data. This decentralises data and allows huge systems interact intelligently.

How has it been tested?: A simulated device, set-up locally on our computer (under Ubuntu in our case), connected to the IoTHub, is generating a steady flow of data points (here: Date, Time, random Temperature, random Humidity). Unfortunately the Python SDK has some troubles under Linux (GitHub Issue), which is why I was not able to use Python and had to put up with node.js and some copy and pasting of mentioned weather data script.
For connecting IoT-devices Microsoft provides libraries to download in either the pip-manager for python or npm for node.js. This makes, in theory, the set-up very simple.
Every device connected to the IoTHub gets a device-ID so that a huge number of devices can be connected via this one platform.
In our exemplary scenario the SimulatedDevice.js and the MessageReceiver.js (connected to the IoTHub) are running, the data is flowing into the IotHub. For forwarding this data a Stream Analytics job is created in Azure. Stream Analytics allows therefore real-time or batch data analytics of events. These events are for example sensor measurements in IoT-devices, or more general any kind of message sent towards the IoTHub.
In Stream Analytics an input source is chosen (SimulatedDevice) and a sink defined. A data sink defines where the data stream is being processed or stored. Functionalities included in the Stream Analytics environment are Anomaly Detection (since it is just a preview for an upcoming function it has not been investigated further yet) or Machine Learning.

In the first example it is connected to Visual BI, a software allowing to set up a dashboard for real-time visualisation. While it can be assumed that there is much more to create than a simple line graph of the temperature is has been only taken into account to test the data stream and the Stream Analytics job. The main goal was to save the data and use it in Machine Learning Studio.

This is where 2 ideas of learning should be mentioned: on-line learning, important in stock market prediction and other fast changing processes, or batch-learning, where the long-time development is much more important. Implementing Machine Learning into the Stream Analytics job would be considered on-line learning. An example that can be seen repeatedly for on-line learning is "sentiment learning" where real-time twitter messages are supposed to show the current sentiment in community. It is less interesting for experimental measurements. For this batch-learning is preferred. It means that big amounts of data is used once-in-a-while to learn and update systems. It can be automatised as a scheduled learning process.
Using batch-learning I can set as Data Input a .csv (or other file formats I upload, or connect a blob storage with the Machine Learning Studio. Again the interwoven functionalities of Azure simplify that process to an incredible small amount of work besides some point-and-click operation.

Conclusion: Can we already deliver a first conclusion on the use of this? It can definitely be said that Machine Learning Studio lacks some functionalities that can be found in other Software such as scikit (Python) and RapidMiner (another proprietary data mining software), but thanks to many set-up simplification in messaging, scheduling and storing data the Azure platform is quite interesting for people who want to connect their data directly to the cloud and even further: want to analyse data on the go with some basic Machine Learning algorithms.

Concerning the "basic ML algorithms": It might be much more functional if R- and Python-scripts have been implemented into Machine Learning Studio. This allows the user to use R's algorithms and in Python to use scikit-learn and TensorFlow and therefore might extend the algorithm selection.
Attached some screenshots of the setup used to test the functionalities:

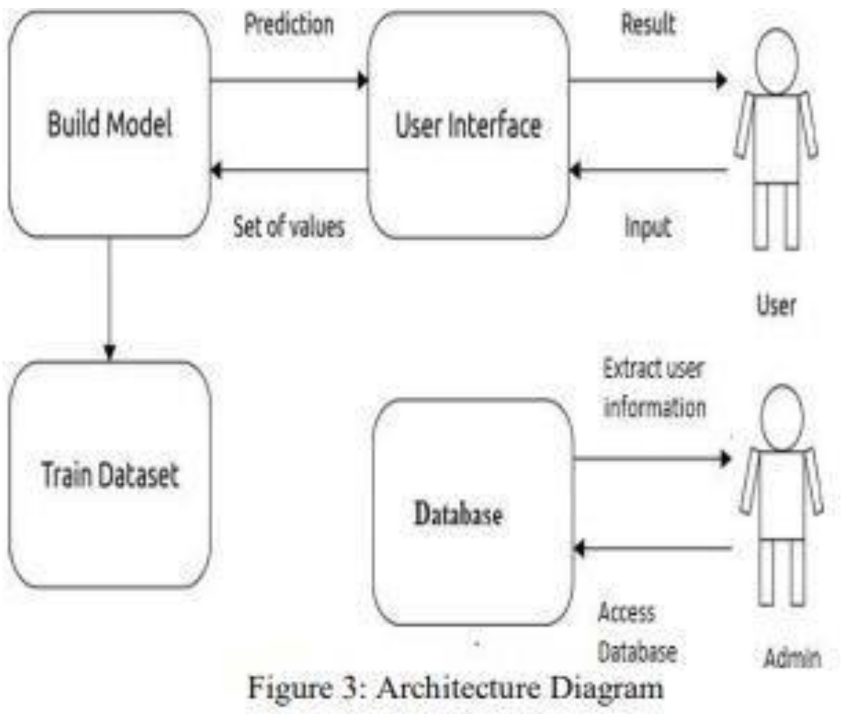
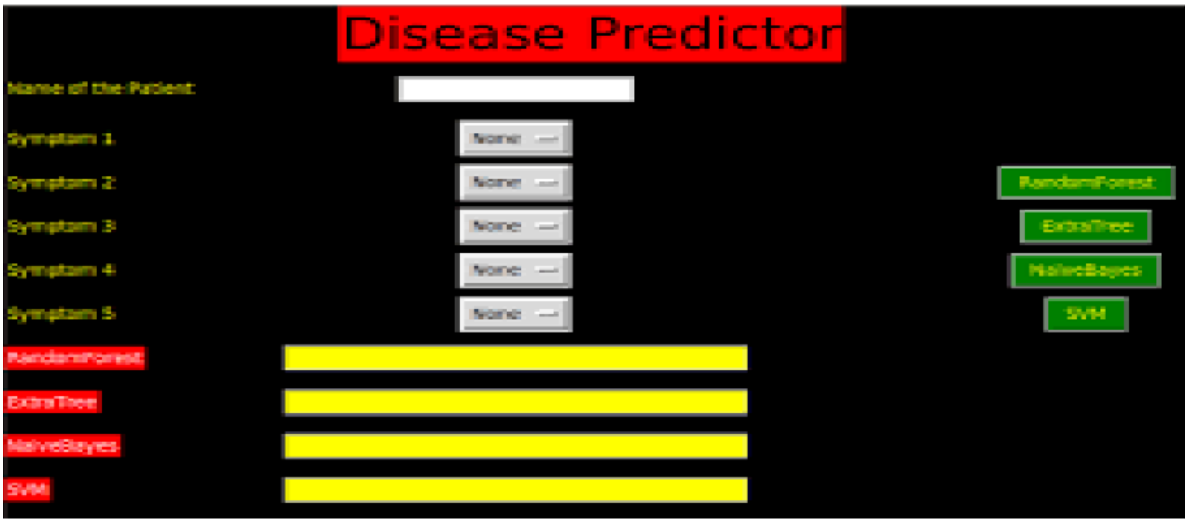


Figure A.1: Real-time data analytics concept


Disease Predictor

Name of the Patient:

Symptom 1 None

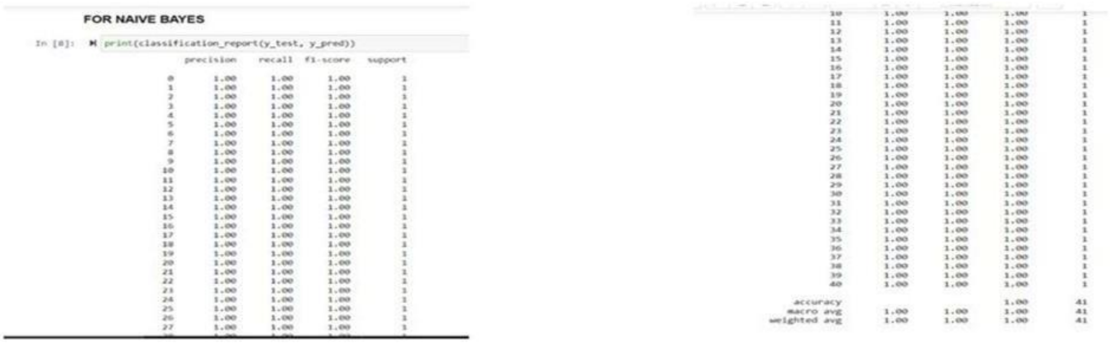
Symptom 2 None

Symptom 3 None

Symptom 4 None

Symptom 5 None

Figure A.2: Simulated sensor generating randomised data.



FOR NAIVE BAYES

```
In [8]:  print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00         1
           4       1.00      1.00      1.00         1
           5       1.00      1.00      1.00         1
           6       1.00      1.00      1.00         1
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         1
           9       1.00      1.00      1.00         1
          10       1.00      1.00      1.00         1
          11       1.00      1.00      1.00         1
          12       1.00      1.00      1.00         1
          13       1.00      1.00      1.00         1
          14       1.00      1.00      1.00         1
          15       1.00      1.00      1.00         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         1
          18       1.00      1.00      1.00         1
          19       1.00      1.00      1.00         1
          20       1.00      1.00      1.00         1
          21       1.00      1.00      1.00         1
          22       1.00      1.00      1.00         1
          23       1.00      1.00      1.00         1
          24       1.00      1.00      1.00         1
          25       1.00      1.00      1.00         1
          26       1.00      1.00      1.00         1
          27       1.00      1.00      1.00         1
```

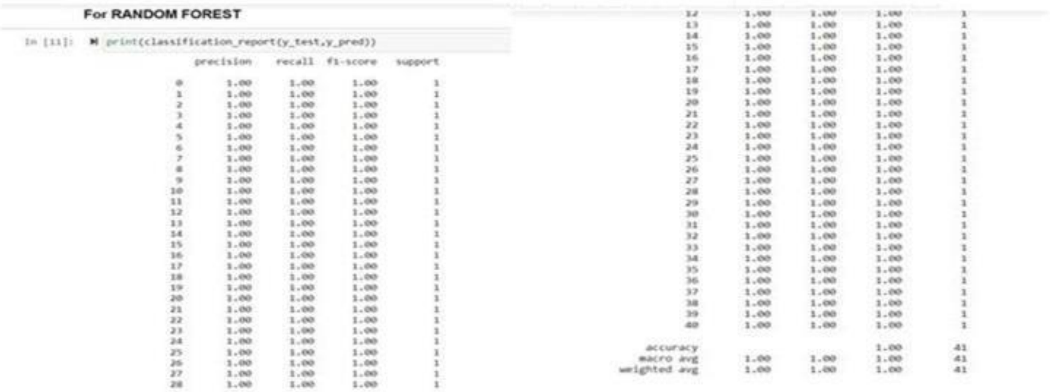
```
          10       1.00      1.00      1.00         1
          11       1.00      1.00      1.00         1
          12       1.00      1.00      1.00         1
          13       1.00      1.00      1.00         1
          14       1.00      1.00      1.00         1
          15       1.00      1.00      1.00         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         1
          18       1.00      1.00      1.00         1
          19       1.00      1.00      1.00         1
          20       1.00      1.00      1.00         1
          21       1.00      1.00      1.00         1
          22       1.00      1.00      1.00         1
          23       1.00      1.00      1.00         1
          24       1.00      1.00      1.00         1
          25       1.00      1.00      1.00         1
          26       1.00      1.00      1.00         1
          27       1.00      1.00      1.00         1
          28       1.00      1.00      1.00         1
          29       1.00      1.00      1.00         1
          30       1.00      1.00      1.00         1
          31       1.00      1.00      1.00         1
          32       1.00      1.00      1.00         1
          33       1.00      1.00      1.00         1
          34       1.00      1.00      1.00         1
          35       1.00      1.00      1.00         1
          36       1.00      1.00      1.00         1
          37       1.00      1.00      1.00         1
          38       1.00      1.00      1.00         1
          39       1.00      1.00      1.00         1
          40       1.00      1.00      1.00         1

    accuracy                           1.00        41
   macro avg       1.00      1.00      1.00        41
weighted avg       1.00      1.00      1.00        41
```

Fig. 4 Classification report Naïve bayes

Figure A.3: IoTHub receiving Data from simulated device.

2. Random Forest Classifier



For RANDOM FOREST

```
In [11]:  print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00         1
           4       1.00      1.00      1.00         1
           5       1.00      1.00      1.00         1
           6       1.00      1.00      1.00         1
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         1
           9       1.00      1.00      1.00         1
          10       1.00      1.00      1.00         1
          11       1.00      1.00      1.00         1
          12       1.00      1.00      1.00         1
          13       1.00      1.00      1.00         1
          14       1.00      1.00      1.00         1
          15       1.00      1.00      1.00         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         1
          18       1.00      1.00      1.00         1
          19       1.00      1.00      1.00         1
          20       1.00      1.00      1.00         1
          21       1.00      1.00      1.00         1
          22       1.00      1.00      1.00         1
          23       1.00      1.00      1.00         1
          24       1.00      1.00      1.00         1
          25       1.00      1.00      1.00         1
          26       1.00      1.00      1.00         1
          27       1.00      1.00      1.00         1
          28       1.00      1.00      1.00         1
```

```
          12       1.00      1.00      1.00         1
          13       1.00      1.00      1.00         1
          14       1.00      1.00      1.00         1
          15       1.00      1.00      1.00         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         1
          18       1.00      1.00      1.00         1
          19       1.00      1.00      1.00         1
          20       1.00      1.00      1.00         1
          21       1.00      1.00      1.00         1
          22       1.00      1.00      1.00         1
          23       1.00      1.00      1.00         1
          24       1.00      1.00      1.00         1
          25       1.00      1.00      1.00         1
          26       1.00      1.00      1.00         1
          27       1.00      1.00      1.00         1
          28       1.00      1.00      1.00         1
          29       1.00      1.00      1.00         1
          30       1.00      1.00      1.00         1
          31       1.00      1.00      1.00         1
          32       1.00      1.00      1.00         1
          33       1.00      1.00      1.00         1
          34       1.00      1.00      1.00         1
          35       1.00      1.00      1.00         1
          36       1.00      1.00      1.00         1
          37       1.00      1.00      1.00         1
          38       1.00      1.00      1.00         1
          39       1.00      1.00      1.00         1
          40       1.00      1.00      1.00         1

    accuracy                           1.00        41
   macro avg       1.00      1.00      1.00        41
weighted avg       1.00      1.00      1.00        41
```

8. Fig. 6 Classification report for random forest

Figure A.4: Import of saved Data from Azure Blob Storage.

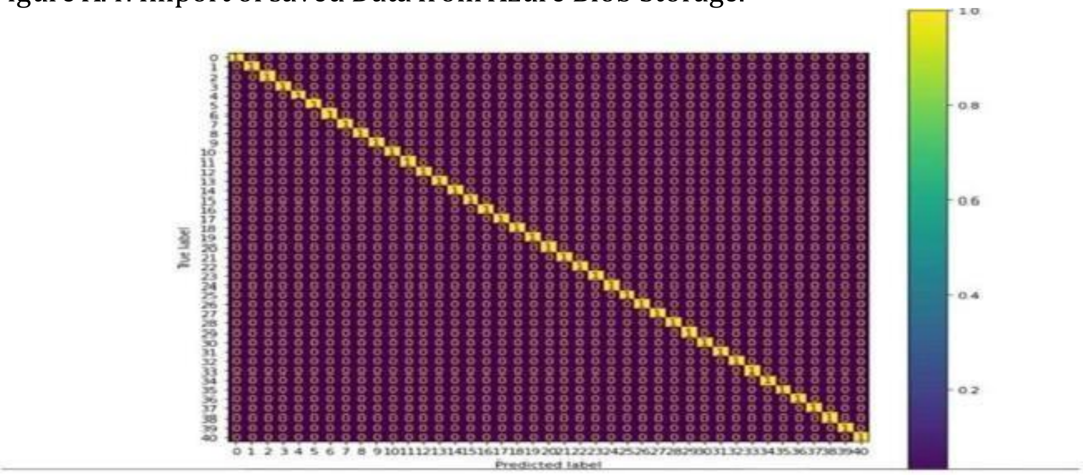


Figure A.5: Real-time visualisation in MS Power BI.