A Weather/Mobility Analysis using Machine Learning

A study about how rain affects ridership in Bergen

Author: Kenneth Apeland
Academic Supervisor: Rogardt Heldal

Master's thesis in Software Engineering at

Department of Informatics,
University of Bergen

Department of Computing, Mathematics and Physics,
Bergen University College

June 2020

Abstract

There are many ways to get from point A to point B, and the way you choose your transport type is directly connected to your global footprint. Citizens choosing to use their car have a high global impact which is why it is important to get citizens to walk, bike, or use public transport instead. When doing so, the cities and public transport companies need to check for different factors that change the number of citizens using public transport. One such factor may be the weather as many of us do not like to go outside when the weather is bad.

In this research, we will analyze the relationship between weather and the number of citizens using public transport using machine learning. We will create three machine learning models and pick the most fitting to be used our backend. This backend will provide data to our frontend containing a prediction about how many passengers there will be the next rush hour and a page where you can manually write hour, day, month, and amount of rain. Using this prediction we can see that there is often an increase when there is a small amount of rain and a decrease when there is lots of rain. One difference we found is that in the summer vacation in July, there is a decrease from the start and it goes back to neutral around 3.8-4.0 millimeters of rain.

Acknowledgements

We would like to extend a special thanks to my talented supervisor Rogardt Heldal from Western Norway University of Applied Sciences, Tom Osnes Svellingen from the MUST team at Bergen municipality and Mikal Fuglestein, a student from University of Bergen. You have always been there for me whenever I had questions or needed guidance. Thanks to the others from MUST team who has helped me with ideas for the thesis.

I would also like the thank all my friends at the Department of Informatics at University of Bergen. Sitting together and helping each other did not go as planned this last semester because of COVID-19, but I have enjoyed all of our online meetings.

Finally, I want to thank my girlfriend and family. Your support and encouragement have been the driving factors for finishing my studies. All the help and advice given has helped me get to where I am today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

There is a trend in the world trying to understand and even change the citizens' mobility patterns due to increasing traffic congestion since the 1950s and environmental reasons. It has become more and more popular to have separate lines for busses, introducing tram lines, bike paths, and walking areas. All this is done to make cities around the world more likable place to live and increase the use of public transport, but it also has a high cost in investment.

The goal for every city is a perfect harmony between every part of the transportation network. Citizens should advance without delay, and the flow of transportation should be well managed without people thinking about it. However, is it enough to only consider the right connections? No, one also needs to understand the behavior of people. Today, this is more and more possible due to all data we collect from sensors put on, for example, roads, bikes, busses, and even people. In this thesis, we will consider whether rain influences the behavior of the citizen.

To consider the weather is essential since there are many climate zones all around the world. What if the weather, such as rain, hinders this change towards more public transport? The climate zones can be small, and there might even be different zones in a small country like Norway. Bergen has had 200 days with rain on average the last 30 years compared to 115 days in Oslo [56].

Bergen is looking to be environmentally friendly and compete to win the European Green Capital Award. To do so, one of the changes they want to enforce is to turn more citizens away from their cars and make them use public transport instead. This would result in less emission produced by driving, which makes Bergen more environmentally friendly and also makes the air quality better for the citizens living here. Bergen will not, however, be able to enforce that change if the citizens do not want to use public transport when it is raining. Predicting the behavior of people will support the move to a more dynamic timetable based on the number of predicted passengers public transport vehicles in use. A more dynamic timetable, or even a booking service, might lead to less traffic on the road, which will lead to less air pollution.

To reduce traffic is not only a local goal but also a global goal, to reach two of the United Nations Sustainable Development Goals. Goal number 11, Sustainable Cities and Communities, tells us that cities and metropolitan areas are powerhouses of economic growth—contributing about 60 percent of global GDP. However, they also account for about 70 percent of global carbon emissions, and over 60 percent of resource use [31]. Goal number 13, Climate Action, presents that the carbon dioxide (CO2) levels and other greenhouse gases in the atmosphere rose to new records in 2019, and 2019 was the second warmest year on record [32].

In this thesis, we will investigate how individuals move in Bergen in terms of weather and mobility data. This research will be done in collaboration with MUST, Mobility Lab for Smart Transport Solutions, in Bergen municipality. They gave us access to their data lake containing data from public transport, biking, and cars. We will analyze how individuals move during the weekdays, both in and out of peak hours and the weekend, and give recommendations about the mobility in Bergen based on our findings.

1.2 Goals and Research Question

In this thesis, the main goal is to analyze whether or not there is a correlation between the number of people using different means of transport depends on the weather and also be able to predict how many individuals who are going to go on and off at different stops. We will conduct this research using the data from the data lake we got access to from collaboration with the MUST project. Furthermore, we will build the code into the existing MUST project cloud service. To accomplish these goals, we will attempt to answer the following research questions:

- How well can we predict ridership on Bergen light rail using machine learning?
- How much correlation is there between rain and the number of passengers on public transport?

- Does rain in the summer have a higher impact on passenger numbers on public transport compared to the rest of the year?

In addition, we have defined two sub-goals. The first is to use different machine learning models to find the best one for our data and the second is to create an easy to understand website that can be ported to MUST.

1.3 Related Work

Research on transportation needs has started all over the world. Researchers want to know the impact of weather and cites want to optimize their transport network for its citizens. Most cities get clogged during rush hour and are researching how to optimize lines so that they can get the transport flowing freely.

H. Asbjørn Aaheim and Karen E. Hauge wrote about the impacts of climate change on travel habits on a national level in Norway. The observations used in this study were average daily wind speed and temperature, and daily precipitation. They found that the response to a given change of climate thereby depends on the level of temperature and precipitation in the city where they live, and the response to the climate scenario depends on how much the climate changes. The report also shows that the response in Bergen is higher than in the other cities because the climate changes to a greater extent [1]. We want to add on the research done on precipitation by scaling down to hourly forecast instead of using the daily levels.

Vanessa et al. studied the effects weather has on human mobility on the west coast of Scotland. They found that rain during the weekend had no key role in travel modes, but heavy rain decreases the use of public transport during the weekdays [5]. We hope to reproduce some of the same results because the west coast of Scotland is similar to Bergen in terms of geography.

Abhishek et al. studied the impact the weather had on urban transit ridership in Manhattan and Bronx in New York City, United States. They looked at temperature, wind, fog, rain, and snow. They found that rain and wind had a higher negative impact on elevated stations compared to underground stations. Thus the elevated stations may benefit from a better designed weather protection features [42]. Bergen does not have any underground station which is why we might not see the same high negative impact on our stations.

Guo et al. studied the impact the weather has on urban transit ridership in Chicago, United States. One of the correlations they found was that rain during the weekend had a key role in the different travel modes in Chicago. The average decrease in daily ridership was 3.13 percent on public transport [42].

Kalkstein et al. studied the effect air masses had on rail transit ridership in Chicago, San Francisco Bay area, and northern New Jersey in the United States. Dry and comfortable days were reported to have significantly higher ridership than days with cool and moist weather with up to 10 percent difference [14].

As shown, it looks like there are differences for each city in how they behave dependent on the weather and transport types. We want Bergen to have its research done in this field to better understand the usage of public transport. In addition, we have yet to find any research done on the relationship between weather and public transport using machine learning. We want to use prior knowledge from the studies above as a baseline for our research.

1.4 Thesis Outline

Chapter 1: Introduction
In Chapter 1 we introduce the background of the problem statements, related work, and our motivation behind why we are looking further into our stated problems. We also present our goals and research questions behind the thesis.

Chapter 2: Background
In Chapter 2 we describe the theoretical background of machine learning, different models, evaluation metrics, and how to get a better model or prediction. Also, we will give you a basic understanding of what a data lake is, and Microsoft Azure.

Chapter 3: Context
In Chapter 3 we introduce MUST and some background about how MUST came to be. We will also talk about public transport in Bergen and give some information about bus and light rail.

Chapter 4: Research Methodology
In Chapter 4 we present the design science research methodology. We will explain our objective of the solution and choice of data that we ended up using.

Chapter 5: System Architecture
In Chapter 5 we introduce relevant information about our system architecture and the technologies used for making the application. This includes the different layers, such as data, business, and presentation layer, and how they exchange

information with each other.

Chapter 6: Analysis
In Chapter 6 we will explain how we will do the analysis of the rain and ridership relationship. We will also explain how to evaluate the website we are going to create.

Chapter 7: Design and Implementation
In Chapter 7 we describe how to prepare our environment and data. This includes defining different input labels we can train on to try to get a better prediction. We will also show how we implemented grid search and our machine learning models together with the hyperparameters.

Chapter 8: Results and Evaluation
In Chapter 8 we introduce our findings from our models and choose the best one. We will use the best one to create generalized and specialized visualizations based on the predictions from the model and present our findings.

Chapter 9: Web Application and Survey
In Chapter 9 we present the web application made for MUST and the survey we made together with the rules for creating a good survey. We will also present the results of the survey.

Chapter 10: Threat To Validity
In Chapter 10 we will talk about the threat to validity and show some of the threats relevant to our thesis. We will also present what we have done to counteract these threats.

Chapter 11: Conclusion
In Chapter 11 we will examine our findings, encapsulate the result, and conclude based on our research questions. We will also give a short briefing about what can be done in future work.

Chapter 2

Background

This chapter contains background on the machine learning and software library we used, and the data lake that we obtain the data from. Keep in mind that it is only the background and we will explain further in detail what we decided to implement in chapter 7, Design and Implementation.

2.1 Machine Learning

The whole concept of machine learning is that you feed the model with data and the data gives you an answer based on the input. It is best described in an online marketplace. With an online marketplace, you want the customers to buy as much as possible. The most intuitive way to do this is to collect as much data as you can from the customers and make personalized recommendations for everyone based on their earlier choices. You can do this manually, but the easier way is to feed the data to a machine learning algorithm and let it give the recommendations. There are many different algorithms to choose from and all of them have different strengths and weaknesses. It depends on the problem you have and what you want to solve. There are three main types of machine learning algorithms: [21].

1. **Supervised learning:** This type of algorithm is used when you have a labeled dataset. This means that for each given dataset, an answer or solution is also given. If you insert a picture of a dog, then you should also tell the algorithm that it is a dog. The algorithm learns by mapping the predicators to the answer, and you will be able to predict the answer afterward. Examples of Supervised Learning: Decision Tree, Random Forest, KNN, and many more.
2. **Unsupervised learning:** In this algorithm, we do not have the answer to our predicators. This type of machine learning is used for clustering data

into different groups based on what we pass into the algorithm. Example of Unsupervised Learning: K-means.

3. **Reinforced learning:** When using this algorithm, the computer is trained to make specific decisions. The machine is exposed to the environment where it will train itself using trial and error. The machine then learns from experience and tries to do the best possible thing in the next iteration. Examples of Reinforced learning: Markov Decision Process and Q learning.

Within these three types of models, you can choose to build a single strong predictive model, or you can build an ensemble of models for some particular learning task. A strong predictive model can be Neural Network where you only have one model you can optimize in different ways, such as making the network deeper or wider. The other type of model is an ensemble of weak learners such as decision trees. This can be done as bagging or boosting. Bagging is where all the weak models predict and the majority decides what it should predict and boosting assigns a set of weights to the models dependent on the learners' error. Extreme Gradient boost is one such model that relies on combining a large number of weak models to obtain a better ensemble prediction [30].

We are going to use supervised learning with single strong predictive models and ensemble models. We would then need to train our model and provide the solution. To be able to test the model accuracy afterward, we also need some test data. We will later present the train-test-split method that will help us split the dataset into two sets, training and test, with a size given.

2.1.1 Machine Learning Libraries

In our work, we decided to use machine learning libraries such as Scikit-learn and Keras as these are well documented and reasonably easy to use.

Scikit-learn

Scikit-learn is a free machine learning library for the programming language named Python. It is one of the most popular machine learning libraries on GitHub, and because it contains most of the machine learning algorithms used today. The project started as a Google Summer of Code project, and then the French Institute for Research in Computer Science and Automation took leadership of it and made the first public release in February 2010 [39]. Scikit-learn also has different preprocessing and evaluation methods to help when making a whole pipeline. Therefore we decided to use this as our main library when we started implementing our models.

Keras

Keras is a high-level neural network API written in Python. It is designed to enable fast experimentation and development and focuses on being user-friendly and modular. Keras was initially developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author maintainer is the Google engineer François Chollet. The user can choose between TensorFlow, Theano, Cognitive Toolkit, or a mix of them as backend depending on what they need in their model [16]. We decided to use Keras as our neural network with TensorFlow as our backend as it has support for running on GPU and a Scikit-learn wrapper.

2.2 Searching For The Optimal Model

Searching for the most fitting machine learning model requires some basic prior understanding of theories such as bias-variance trade-off and regularizations. We will also present cross-validation and hyperparameter optimization used to find better models.

2.2.1 Bias-Variance Trade-off

It is hard to create the perfect model because a model should be sensitive enough to accurately capture the key patterns in the training data and be generalized enough to work well on unseen data. We will always have a risk of overfitting when trying to solve the first point, especially if we have noisy or unrepresentative data. On the other side, we will always have a chance of generalizing too much and end up not getting the important regularities [47]. To better understand how to create a model hitting the trade-off with supervised learning, we need to look at the bias-variance trade-off.

Bias

The model is underfitted or has high bias if the model accuracy is low on both the training and test dataset. There are two important reasons for overfitting/bias. The first one is not including the right features in the dataset and the second is not picking the right parameters for the models [47].

Variance

You can get a high variance or overfit the model if you have high variation in the training set or if you choose parameters wrongly for your model to get higher accuracy when you are training. You can try to reduce the number of features or use a regularization method that will reduce the extent of the features. You can also try to increase the dataset so that you can feed the model more data [47].

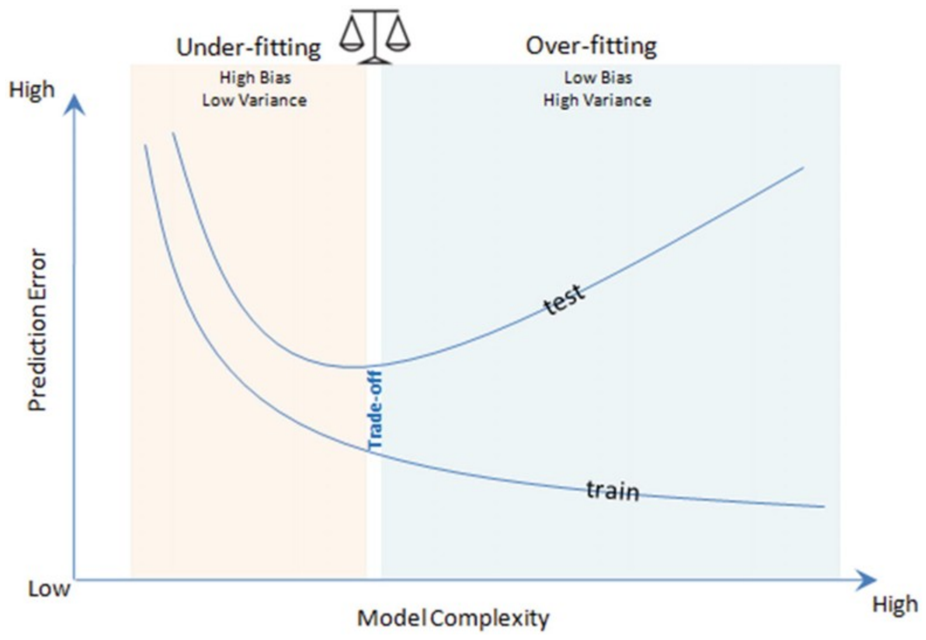


Figure 2.1: Visualization of the correlation between bias and variance
Credit: Manohar Swamynathan [47] (Retrieved 30/4/2020)

2.2.2 Regularization

Regularization in machine learning is important to prevent overfitting. Looking at it in mathematical terms, it adds a regularization term to prevent the coefficients to fit so perfectly and end up overfitting. The difference between the L1 (Lasso Regression) and L2 (Ridge Regression) is that L1 adds the *“absolute value of magnitude”* of coefficients as a penalty term to the loss function and L2 adds the *“squared magnitude”* of coefficients as a penalty term to the loss function [29].

L1 regularization on a loss function

$$L(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^t)^2 + \lambda \sum_{i=1}^{t} |w_i|$$

L2 regularization on a loss function

$$L(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^t)^2 + \lambda \sum_{i=1}^{t} w_i^2$$

In both L1 and L2, there is a λ. If we set the λ to zero, then the regularization would end up as zero and we would only count the loss function and we might end up with an overfitted model. If we set the λ to a high number, then the weights will be too heavy, and it will lead to underfitting.

We need to find a good balance between overfitting and underfitting when we are creating our models together with regularization. There are times when you can use a high number on both L1 and L2, and other times where you cannot use any of them.

2.2.3 Cross Validation

Cross-validation in machine learning is used to evaluate machine learning models on a limited data sample. The general concept is that you specify the number of folds you want to have and that will also specify how many times you want to train and test on the data. Let us say that you will try 10-fold. It will in the first iteration use 0-99 as test data and 100-999 as training data. It will use 100-199 as test data and 0-99 and 200-999 as training data in the second iteration. It will do this 10 times with 10 percent of the data as a test every time and end up with testing on 100 percent on the data in total [19].

The number of folds is often determined by the amount of data you have. You might need to use a lower number of folds if you have a small amount of data, or you can use a high number if you have a large amount of data. We will decide a number fitting for us when we create our models in chapter 7.4.

2.2.4 Hyperparameter Optimization

There exists a set of hyperparameters for every given machine learning model. We will write about these hyperparameters for every algorithm later in this chapter. There are four types of hyperparameter optimization methods [3]:

- **Manual:** Select the hyperparameters based on guessing, train the model, and score with test data. Repeat the process until you are satisfied with the result.
- **Random search:** Set up a grid of hyperparameters with values and select random combinations to train the model and score with test data. This method will take the number of combinations as input.
- **Grid search:** Set up a grid of hyperparameters with values and train and test the model on every combination.

- **Automated tuning:** Using methods such as gradient descent, Bayesian Optimization, or evolutionary algorithms to conduct a guided search for the best hyperparameters.

Manual and random search can result in reaching a local minimum if they are far from the global optimal solution, and this is why we decided to use grid search as this will always result in the global minimum if you give it enough parameters. The time elapsed when using grid search will increase exponentially with the number of different values inside the different parameters, but will be worth it in terms of increased performance of the model [6].

2.2.5 Methods From Libraries

As mentioned earlier, the data for training and testing often comes from the same dataset. This means that we have to split the data into two parts to make the model test on unseen data. *train-test-split* is a method from Scikit-learn that makes it easy for programmers to split the data into 2 different sets with a random state. You can manually set this random state to get a consistent result when searching for the best parameters with a method like grid search. You can also specify the test size dependent on what you see fit [40].

There are two methods from Scikit-learn helping with the encoding of the data, named *Label Encoder* and *One Hot Encoder*. Both of these methods could help the models understand the data better and give a better prediction. The Label Encoder will transform every categorical value into a numerical value, and it will give every category the same number. We have applied the Label Encoder on the *Day* column in the example in table 2.1.

$$\begin{array}{c} \begin{array}{cc} \text{Day} & \text{Transport} \\ \text{(Text)} & \text{Type} \end{array} \\ \begin{bmatrix} \textit{Monday} & \textit{Bus} \\ \textit{Tuesday} & \textit{Bike} \\ \textit{Friday} & \textit{Bus} \\ \textit{Monday} & \textit{Car} \end{bmatrix} \end{array} = \begin{array}{c} \begin{array}{ccc} \text{Day} & \text{Day} & \text{Transport} \\ \text{(Text)} & \text{(Numerical)} & \text{Type} \end{array} \\ \begin{bmatrix} \textit{Monday} & 1 & \textit{Bus} \\ \textit{Tuesday} & 2 & \textit{Bike} \\ \textit{Friday} & 3 & \textit{Bus} \\ \textit{Monday} & 1 & \textit{Car} \end{bmatrix} \end{array}$$

Table 2.1: Showing the transformation with Label Encoder

The problem with Label Encoder is that some models will look at the average in one column. This means that the average between Monday and Friday is Tuesday ($1 + 3 = 4/2 = 2$). The One Hot Encoder fixes this problem by taking the categorical or numerical values and transform them into their columns. The example in table 2.2 shows an One Hot Encoding on the column named *Day*. The number *1* shows that the feature is existent and *0* shows that the feature is nonexistent.

$$\begin{array}{c} \begin{array}{cc} \text{Day} & \text{Transport} \\ \text{(Text)} & \text{Type} \end{array} \\ \left[\begin{array}{cc} \textit{Monday} & \textit{Bus} \\ \textit{Tuesday} & \textit{Bike} \\ \textit{Friday} & \textit{Bus} \\ \textit{Monday} & \textit{Car} \end{array}\right] \end{array} = \begin{array}{c} \begin{array}{cccc} \text{Monday} & \text{Tuesday} & \text{Friday} & \text{Transport} \\ & & & \text{Type} \end{array} \\ \left[\begin{array}{cccc} 1 & 0 & 0 & \textit{Bus} \\ 0 & 1 & 0 & \textit{Bike} \\ 0 & 0 & 1 & \textit{Bus} \\ 1 & 0 & 0 & \textit{Car} \end{array}\right] \end{array}$$

Table 2.2: Showing the transformation with One Hot Encoder

2.3 Choosing An Algorithm

Even to this day, the no free lunch theorem still stands. This theorem states that there is no best machine learning algorithm for all problems [51]. Therefore, when working with machine learning, you should always try different algorithms to find the best one for your exact problem. Seeing that we chose to predict the number of citizens using a stop, we knew we had a linear regression problem. We also knew that we had multiple inputs to the model which means that we needed to find algorithms with support for what is known as multiple linear regression.

After looking at what others in the machine learning community used for multiple linear regression models, we decided that we wanted to try one simple model first. In this case that would be the Linear Regression model. We wanted to look at two more advanced models after looking at the simple model and ended up with Neural Network and Extreme Gradient Boosting.

2.3.1 Linear Regression

Linear regression is a regression algorithm that is a statistical technique used to model the relationship between two variables and understanding how they contribute to a particular outcome together. It is a simple model that is widely studied as it is widely used in statistics. The linear regression model creates a function predicting the output based on the input parameters in numerical form [52]. We will use a multiple linear regression algorithm which will produce a prediction with the formula below:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \epsilon$$

The error ϵ is added to the function as a statistical error. This statistical error will account for the failure of the model to fit the data exactly. The x_i is the value coming from our data-points for each parameter. β_0 is a constant and $\beta_1 - 4$ is the slope coefficients. To find the β-values, we will need a loss function called ordinary least squares [24], not unlike the functions explained in section 2.4. This function will optimize the line (the value y), meaning it will update the constant and slope coefficients in our function from earlier.

The ordinary least squares linear regression does not have any hyper-parameters to optimize the fit to the data. You can only add or remove features, or add or remove some data in the preprocessing. You have to use another model if this still does not give a satisfactory result.

2.3.2 Neural Net

A neural network works the same way as your brain as it consists of different layers and artificial neurons inside those layers. These layers are completely independent of each other, meaning that it works out if it should fire or not by its input and weights regardless of what the other neurons are doing. The only thing the neurons share is the inputs as every neuron sees all the inputs in the network [21]. A neural network is an iterative optimization algorithm used to find the best result with the minima of a curve (gradient descent). Learning the network requires three layers and a learning round is as follow:

1. **Input layer:** This layer will have a set of neurons where each neuron is linked to each feature in our dataset. This layer will pass it on to the hidden layer.
2. **Hidden layer(s):** There is a set of n number of neurons where each neuron has its weight assigned to it expressing the importance of the respective input to the output.
    (a) The neuron will take the sum of all inputs from the previous layers and multiply the inputs and weights together and add bias to create a numerical value. The formula is as follow: $value = b + \sum_{i=1}^{n} x_i w_i$.
    (b) Apply said value to the activation of your choosing such as relu: $next = max(0, value)$.
    (c) Pass it to the output layer.

There can be as many layers as you want, but the last hidden layer will pass the data to the output layer.

3. **Output layer:** The output layer does the same as the hidden layers, but it will return the outcome as a value or a classification. When it returns, it will also compute the error based on the given actual value connected to that input.

When one round of forward learning, called forward-propagation, is done, it will start the backward-propagation. This means that it will update the weights on its way back to the input layer from the output layers given with the given error and learning rate. The learning rate tells the network how much the weights should update each round. The weights to a neuron will not update if the values are correct, as it will only update the once who activated on the wrong data.

The algorithm will do this X number of times and the neural network will get closer and closer to the correct answers. This is how it learns, but you can also train it too many times and get overfitting. Overfitting will lead to incorrect answers when predicting unseen parameters. You can apply what is called Dropout to the hidden layers which will randomly select neurons with a given probability and disable them for that run and as a result give a more generalized model. This can help your score when testing on unseen data [21].

Hyperparameters

A neural network has many different parameters to optimize, both in terms of getting better accuracy and in terms of speed [15].

- **activation:** Normally the activation functions can reflect the accuracy, however with a regression model we need to use *Relu* and *Linear*.
- **number of layers:** A too shallow network will result in worse accuracy. The goal is to keep adding more layers until the test error stops improving.
- **learning_rate:** A low learning rate slows down the process, but converges smoothly. A high learning rate will give higher speeds, but you might not be able to hit the bottom of the gradient descent.
- **epochs:** Epochs is the number of times the network will go through the data. This means that when the network has gone through all the data one time, it has gone through one epoch [48]. You should have a higher epochs number when you have a lower learning rate to end up in the minima of the curve.
- **batch_size:** The batch size is the number of subsamples given to the network from the data as one group.

- **dropout:** Dropout will as mentioned make the model less vulnerable to overfitting. Each training example is forward propagated, but the dropout randomly keeps the outputs of each layer with probability p [13]. A probability inserted too low has minimal effect and a value too high results in under-learning by the network.
- **momentum:** Momentum is a coefficient that is added to the update equation in the backward-propagation and it increases the speed of the convergence [9].

2.3.3 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It uses a machine learning technique named Gradient Boosting. Gradient boosting produces a prediction model with an ensemble of weak prediction models. In XGBoosts case, this is an ensemble of decision trees. XGBoost creates a model with parallel trees that learns and predicts in a fast and accurate way [54].

How it works

We must first understand how Gradient Boosting works before we move on to Extreme Gradient Boosting. A Gradient Boosting model is a model based on Boosting with added gradient descent. The common ensemble models strictly rely on simply averaging the models in the ensemble, however, the boosting methods are based on a different strategy of ensemble formation. The main idea of boosting is to add new models to the ensemble sequentially. At each particular iteration, a new weak, base-learner model is trained concerning the error of the whole ensemble learned so far [30].

The gradient is used to minimize the loss function, similar to how a neural network uses the gradient descent function to optimize the weights. The weak learners are built each round and their predictions are compared to the expected outcome. The difference between the prediction and the expected answer gives us the error rate of the model. The model will now use the error to calculate the gradient. The gradient is the derivative of the loss function associated with the ensemble. You can use the loss function you want, but the most classic is the squared error loss function. The model will then use the gradient to find the direction in which it will reduce the error as much as possible for the next round of training by following the gradient.

The name Extreme Gradient Boosting refers to the engineering goal to push the limit of computations resources for boosted tree algorithms [49]. The word extreme means that it is a specific implementation of the Gradient Boosting method which uses more accurate approximations to find the best trees. It uses a few handy tricks to make it more successful with structured data:

- **Second-order gradients:** It uses the loss function based on the second partial derivatives which give the method more information about the direction of the gradients and how to get the minimum of the loss function. While the regular gradient boosting uses the loss function as is, the extreme gradient uses a descent based on the approximation which is coming from the second-order derivative.
- **Regularization:** Extreme Gradient Boosting uses an advanced regularization, both L1 and L2, which improves model generalization.
- **Parallelized:** Training is very fast and can be parallelized or distributed across clusters.

The list under describes accurately how XGBoost works [28]:

1. For each descriptor,
   - Sort the numbers
   - Scan the best splitting point (highest gain)
2. Choose the descriptor with the best splitting point that optimizes the training objective
3. Continue splitting (as in (1) and (2)) until the specified maximum tree depth is reached
4. Assign prediction score to the leaves and prune all negative nodes (nodes with negative gains) in a bottom-up order
5. Repeat the above steps in an additive manner until the specified number of rounds (trees K) or early stopping is reached.

Hyperparameters

There are two different places to optimize the parameters, and that is on the model itself and when doing cross-validation. Most of the parameter optimization on the model is to get better predictions. The list under shows which parameters we used to optimize the model [53]:

- **learning_rate:** Since we have a gradient descent method, we can specify the learning rate with the parameter between 0 and 1. The learning rate tells us the size of the steps we are going to take down the descent of the gradient. Shrinking the learning rate makes the boosting more conservative which will prevent overfitting, but it will, of course, take longer to run as it needs more booster iterations to get to the bottom of the gradient.

- **colsample_bytree:** There are three types of methods you can choose from when subsampling the columns.
  - *Colsample_bynode* is the subsample ratio of columns for each node (split). Subsampling occurs once every time a new split is evaluated. Columns are subsampled from the set of columns chosen for the current level.
  - *Colsample_bylevel* is the subsample ratio of columns for each level. Subsampling occurs once for every new depth level reached in a tree. Columns are subsampled from the set of columns chosen for the current tree.
  - *Colsample_bytree* is the subsample ratio of columns when constructing each tree. Subsampling will then occur once for every tree constructed.
- **max_depth:** Max depth specifies the maximum depth of a tree. The default is 6 and increasing this value will make the model more complex and more likely to overfit.
- **alpha:** Specifies the L1 regularization. Enabling the Alpha parameter will try to pull the weights to zero as talked about earlier and increasing this value will make the model more conservative.

The parameters in the list below shows which parameters we used for our cross-validation:

- **num_boost_round:** One boosting iteration is the same as creating one tree. The number you specify will decide the maximum number of trees the algorithm can create. Fewer boosting trees are required with increased tree depth, however, this will also result in a higher probability to overfit.
- **early_stopping_rounds:** Specifying a number here enables early stopping. The model needs to improve at least once in every x-number of rounds to continue training. The last entry in the evaluation history will represent the best iteration.
- **nfold:** The number you insert here will be the number used for cross-validation as presented in subsection 2.2.3.
- **metrics:** This parameter specifies the evaluation metric to be watched in cross-validation. We ended up using the mean square error presented in chapter 2.4.

2.4 Evaluating Regression Models

The evaluation of a regression model is not done with the same principles as the evaluation of a classification model. A classification model has different metrics you can look at, such as accuracy, precision, and recall. Accuracy tells you the ratio of the number of correct predictions to the total number of input samples, precision is the number of correct positive results divided by the number of positive results predicted by the classifier and recall is the number of correct positive results divided by the number of samples that should have been identified as positive. You also have a ROC curve which plots True Positive Rate and False Positive Rate [21].

There is no possibility of using any of these metrics in a linear regression model because we only have our predicted values which are a number, compared to a classification. We can only evaluate it based on mathematical error functions:

Mean Square Error

Mean Squared Error is one of the most preferred metrics for regression problems. It is calculated by taking the average of the square of the difference between the actual and predicted values of the data. When taking said square, it penalizes even a small error which could lead to over-estimation of how bad the model is. It is preferred over other metrics because it is differentiable and hence it can be optimized better [23].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Square Error

Root Mean Square Error represents the sample standard deviation of the determined residuals. They can be positive or negative as the predicted value under or overestimates the actual value. Squaring the residuals, averaging the squares, and taking the square root gives us the root mean square error [11].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Mean Absolute Error

A little less used method is the Mean Absolute Error which does not penalize the errors as extremely as Mean Squared Error. Mean Absolute Error takes the absolute difference between the actual output and the predicted output. This means that an error of 10 is twice as much as an error of 5 [23].

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Mean Absolute Percentage Error

The mean absolute percentage error formula uses the absolute difference between the actual output and the predicted output to compute a percentage score as output:

$$MAPE = (\frac{1}{n} \sum_{i=1}^{n} |\frac{|y_i - \hat{y}_i|}{y_i}|) * 100$$

Take the sum of all the absolute values of the difference between the predicted value and the actual value divided by the sum of all the actual values and multiply that by 100. A percentage closer to 100 would mean a better model.

2.5 Azure

Azure is Microsoft's cloud-computing platform for planning, deploying, and managing code and data. Cloud services can be grouped by three different types of services and Azure have tools in every group [35]:

1. **Software as a service (SaaS):** Microsoft uses Office 365 as a SaaS and connects it to the Azure infrastructure. This provides customers with access to Microsoft's top productivity tools without having to implement and maintain significant on-premises infrastructure.

2. **Infrastructure as a service (IaaS):** Azure gives you the possibility to spin up virtual machines and virtual networks, Azure storage solutions, and Azure recovery services all in the web for easy deployment.

3. **Platform as a service (PaaS):** The technologies used for PaaS is Azure SQL Database and the Azure websites.

Azure has a hybrid cloud solution called Azure Stack for those who have extra limitations in terms of privacy. This hybrid solution means that you can have some of the server stacks in your buildings with your hardware for the information that should be extra protected, and the rest can run in the cloud.

As a big web service, it also has a machine learning platform built-in. The user can decide to use the pre-made templates for different models or you can upload your python code and run it. This means that a non-technical person can point and click for some machine learning tasks and programmers can do everything from scratch and import their code. We will opt to do the latter as this gives us more flexibility in terms of picking models and optimize them.

2.6 Data Lake

The data lake came to be because of the economics of big data. Handling and storing big data can be expensive, but the use of data lake can cut the cost from 20 to 50 times compared to a data warehouse because of the agile underlying technologies that typically supports the data lake [38]. In essence, a data lake is a data respiratory where all data in an enterprise i.e. structured, semi-structured, unstructured data + binary data are stored altogether regardless of types, format, or structure. The understanding of the data nature is delegated to the data consumer at the time of data retrieval (i.e. query time). When data are retrieved, the user will transform that data according to the parts of the enterprise to acquire business insight [17].

Comparison	**Data Warehouse**	**Data Lake**
Data	Structured, processed data	Structured/ semi-structured, unstructured data, raw data, unprocessed data
Processing	Schema-on-write	Schema-on-read
Storage	Expensive, reliable	Low cost storage
Agility	Less agile, fixed configuration	High agility, flexible configuration
Security	Matured	Maturing
Users	Business professional	Data Scientists

Table 2.3: Comparison of data warehouse and lake [17]

A data lake works well for MUSTs' vision. They get data from multiple companies and stores them in different stages. There's *Raw* for data in its raw form such as unstructured data, *Archive* for managed and filtered data and *Staged* for structured, but not yet processed data. Figure 2.2 shows that MUST get data from different companies and insert them into their data lake. After doing so they will call the query processing from the different applications they own.

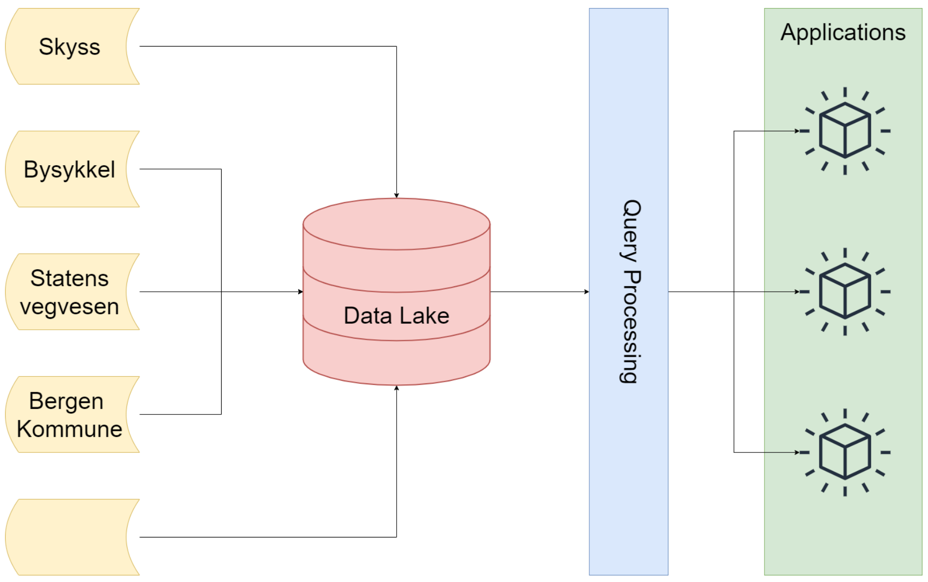


Figure 2.2: Getting data from sources into applications, MUST

Chapter 3

Context

The purpose of this thesis is to produce data that will support change in public transport in the future to better utilize it on rainy days. In this section, we will first describe the organization MUST that drive mobility changes in Bergen follow by some of the partners in MUST. Since this thesis is done in collaboration with MUST, it is essential to describe this organization and its key partner that will use our data. We will start with MUST followed by the partner.

3.1 MUST

MUST is Bergens' Mobility laboratory for the development of smart transport solutions. Hordaland county council at Skyss participated in a national competition with Bergen Municipality with proposals to create a mobile mobility laboratory for the development of smart transport solutions - a MUST [27]. One wanted to establish such a laboratory that will be a catalyst for technology, business, and social development linked to mobility and transport. Here you can, for example, test new solutions and look at how one can best get to the good cooperation between the various actors and modes of transport. Hordaland won 12.5 million NOK in the competition and has pledged to double the winnings so that the project will receive 25 million NOK over six years [43].

The official opening was 27. November 2018 [26] and MUST were divided into three parts. InnoLab contributes to idea development and finding partners, DataLab which helps to verify the concept against data and LivingLab which helps to test the concept. Together they all have projects to help meet the UN's sustainability goals, standardization, scalable architecture, and give better mobility [25]. They made a data lake in Microsoft Azure together with some IT consultants from the consultant firm Bouvet [4]. The data lake contains data about public transport from Skyss, biking, and cars from Norwegian Public

Roads Administration and city bikes from Bergen City Bike. This data lake is the one we used in our research.

3.2 Public Transport

The company responsible for public transport in Bergen is called Skyss. They were established in November 2007 and are owned by Vestland county municipality. The public transport companies Skyss and Kringom in the old county have merged organizationally and are now mobility units in the county of Vestland as of January 1. 2020 [45]. Skyss manages public transport such as bus, trolleybus, light rail, ferry and express boat in Bergen. It is responsible for route planning, awarding of contracts, monitoring the quality of the route offer, sales channels and ticketing, real-time system, customer center, marketing and information, and for further development of the public offering [44]. We will focus on bus and light rail because they are the most used and both of them drive within Bergen city center. The light rail line is the most popular in Bergen, however, the number of people using the bus every day is much higher in total.

3.2.1 Bus

The bus is one of the oldest and most common modes of transportation. It is used both inside a city and as transportation between cities. It has the advantage of being affordable and convenient if there are bus lanes or no traffic on the road. You can also buy the buses you think are the best for your city. You can customize how long they are, and the seating and standing situation. Long-distance and old busses will have many seats, few standing places, and uses fossil fuel. There are some electric busses used within a city using an electric line above, just like a tram, with less seating and more standing area to get more people on board. Many of the new city busses have battery packs instead to save the cost of building a new track. The electric buses are used to lower the emissions and get better air quality within the cities.

3.2.2 Light rail

Tram usually goes a short distance within the city center and the train usually go long distance outside the city center. The light rail is a mix of both of these types of transportation. It has enough capacity to transport many individuals inside the city center, while also having a high top speed to travel further when free from traffic. The easiest comparison is a subway, only that the light rail goes above ground level instead of under. A light rail has a much higher upfront cost compared to a bus because it should have separate lanes free from other traffic and it needs power lines.

The light rail in Bergen does not have the highest number of seats, however, there should be enough for up to 5 times as many individuals who can stand comfortably. The light rail goes from Byparken inside the city center to Bergen Airport located at Flesland as shown in figure 3.1. It had 18.655.000 passengers in 2019 which is over 51.000 every day [8]. This means that the light rail moves the same number as residents in Bergen every 5-6 days [46].

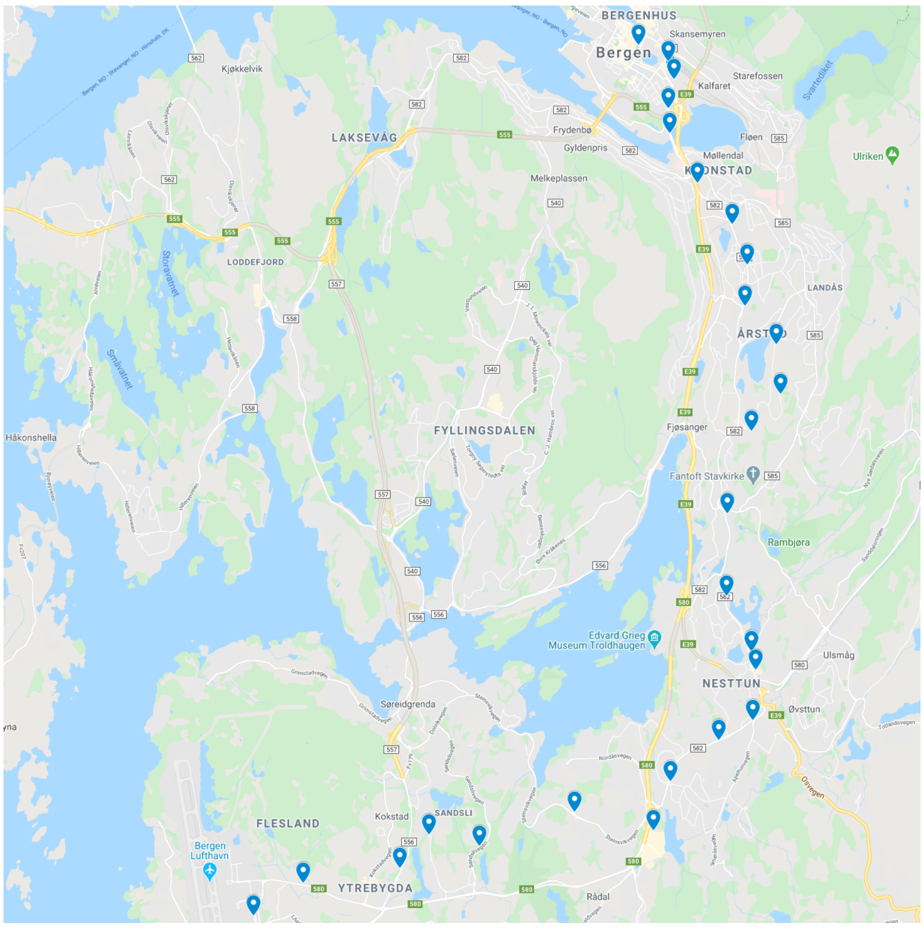


Figure 3.1: Stops on Bergen light rail
Credit: Google Maps (Retrieved 19/3/2020)

Chapter 4

Research Methodology

This chapter contains information about the project methodology design science research. The methodology helped us to structure the research done in this thesis and gives us an understanding of how to structure and sample our data.

4.1 Design Science Research

We opted to use design science as our research methodology. Design science research is a type of method that is set up and used when the goal is an artifact or proposal. The research based on design science is well fitted to be used within a collaboration project between academic and industry when an artifact will be built as it aims to study and research the artifact and its behaviors. Hevner et al. state that design science is used to predict or explain phenomena concerning the artifact's use (intention to use), perceived usefulness, and impact on individuals and organizations [10].

Design science addresses research through the building and evaluation of artifacts designed to meet the identified business need. This means that the goal of design science research is utility. Since design science is a problem-solving process, it also gives us seven guidelines to follow to increase the chances of success. It requires the creation of an innovative purposeful artifact (guideline 1) for a specific domain (guideline 2). Because the artifact is purposeful, it must yield utility for that specific problem. The evaluation of the artifact is crucial (guideline 3) and must be done thoroughly. The artifact needs to be innovative which means that you should solve an unsolved problem or solve a known problem in a more effective or efficient manner (guideline 4). The artifact must be strictly defined, formally represented, and internally consistent (guideline 5). The progress where it is created incorporates or enables a search process by which a problem space is constructed, and a mechanism posed or enacted to find an effective solution (guideline 6). The results of the design science research must be communicated effectively (guideline 7) to both a technical audience and to a managerial audience [10].

There are some differences between the different sets of frameworks made to produce a successful artifact. Peffers et al. present a framework for using design science research for use in information system research. The framework is formed by six steps [34]:

1. **Problem identification and motivation:** Define the specific research problem and justify the value of a solution. Justifying the value of a solution accomplishes two things: it motivates the researcher and the audience of the research to pursue the solution and to accept the results and it helps to understand the reasoning associated with the researcher's understanding of the problem. This is explained in chapter 1.4 where we talked about motivation, goals and related work, chapter 2.6 with the background and chapter 3.2.2 with context.

2. **Define the objectives for a solution:** Infer the objectives of a solution from the problem definition and knowledge of what is possible and feasible. The objectives can be quantitative, such as terms in which a desirable solution would be better than current ones, or qualitative, such as a description of how a new artifact is expected to support solutions to problems not previously addressed. The objective of the solution will be presented in this chapter.

3. **Design and development:** Create the artifact. Such artifacts are potentially constructs, models, methods, or instantiations or some new "properties of technical, social, and/or informational resources". The abstract idea of a design research artifact can be any designed object in which a research contribution is embedded in the design. This activity includes determining the artifact's desired functionality and its architecture and then creating the actual artifact. The artifact in our case will be the machine learning model with a website as we will present in chapter 7.

4. **Demonstration:** Demonstrate the use of the artifact to solve one or more instances of the problem. This could involve its use in experimentation, simulation, case study, proof, or other appropriate activity. A demonstration has been done continuously for MUST and we will present it in this thesis in chapter 8.

5. **Evaluation:** Observe and measure how well the artifact supports a solution to the problem. This activity involves comparing the objectives of a solution to actual observed results from the use of the artifact in the demonstration. We will also create a survey in chapter 9 and see how the

answers there corresponds to our artifact. The evaluation and discussion about how well the artifact works will be done in chapter 8.

6. **Communication:** Communicate the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness to researchers and other relevant audiences such as practicing professionals, when appropriate. Communication has been helpful for this thesis because of all the new perspectives and ideas which have emerged, and this thesis in it-self is a communication channel to the academia. We have also had weekly meetings with MUST and presentations in front of developers from Skyss and The Norwegian Public Roads Administration to further strengthen the research done.

4.2 Objective

The objective of the solution will be divided into two parts. The first part is the artifact containing the machine learning together with the website which is expected to support the solution to a problem not previously addressed. The second part will be the analysis using a quantitative method. The quantitative method is used when you want to measure something and the result is often shown with numbers, graphs, and tables which makes it well fitted for answering questions about the correlation of rain and public transport use [41].

4.3 Data Sources

As stated earlier, we had many different data sources inside the Data Lake owned by MUST. This is the data in the lake we used to research our thesis:

- **Public transport:** Data from Skyss from August 2018 to April 2019. We used a few tables to bind the data together, but the most important table contains the direction, trip status, on, off, trip key, stop key, link key, time, date, route, from and to. This table shows us information about the number of residents going on and off at every stop on every line. This results in a 25.009.048 lines long file which was just over 3.4 Gigabyte big with data from August 2018 to April 2019.

  We got the data from the rest of 2019 in late April 2020 because Skyss were working on creating an API that MUST could connect to and there was a new standard for naming the fields in the data. The new data only has data per hour on each line for each stop instead of one entry for every exact transport type like the old data had. This was no issue for us since we only needed the number of passengers per line per hour. The new file with data from April 2019 to December 2019 does still contains 16.746.132 lines and is just under 2.5 Gigabyte big.

- **City Bike:** This data is from Bergen City Bike. Bergen City Bike has an API which means that the data is imported to the data lake every day. This table contains duration, start station id, start station name, start station description, start station latitude, start station longitude, end station id, end station name, end station description, end station latitude, end station longitude, start time local, end time local. This file is 1.039.001 lines long and just over 200 Megabyte big and contains data from August 2018 to January 2020.

- **Bike:** Data from Bergen municipality. There are counters put up within the city to count how many bikers there are. This data contains the date and hour, what station it is, and how many persons passed within every hour. The file is 66 Megabytes big and contains 372.915 lines from August 2018 to January 2019.

- **Road data:** Vehicle data from The Norwegian Public Roads Administration. This data contains the name of the point counting, date and hour, how many passed, and how many with unspecified length. It had 4.417.802 entries in the file and just under 800 Megabyte when containing data from August 2018 to January 2020

In addition to these data sources, we also downloaded data from the Norwegian Climate Service Center [18] to get the official weather data from a weather station on Florida, Bergen. We could get every type of weather data, but settled with a file containing millimeter of rain, wind, and temperature per hour from August 2018 to December 2019.

Looking at the correlation between the rain and the public transport for the whole city gave us a holistic view, however, there can be differences within a city. The citizens' travel habits may depend on where they live in Bergen and what they use the public transport for. That is why we assume there will be differences between citizens inside the city center and the rural areas. The citizens in rural areas might need a few bus or light rain changes to get to the city center, but those who live closer may not need to change.

We decided to focus on a few stops, and this resulted in also just looking at the passenger numbers on the light rail. Byparken was the first place that came to our mind because it is the most central one and many citizens go on and of here since it is the end of the line. We also added Nonneseter which is stop number two on the line. We did this because we know from personal experience that many go on and off here because they do not want to walk 500 meters between these central stops. However, we do not know if more passengers are using the light rail between these two stops when it is raining or if the amount is constant.

We picked one stop outside the city center to see if there were any difference from the city center and ended up with Kronstad. Kronstad is the stop closest to the Western Norway University of Applied Science. This means that we can check if the number of students traveling to and from the campus at the Western Norway University of Applied Science when it is raining is the same as when it is not.

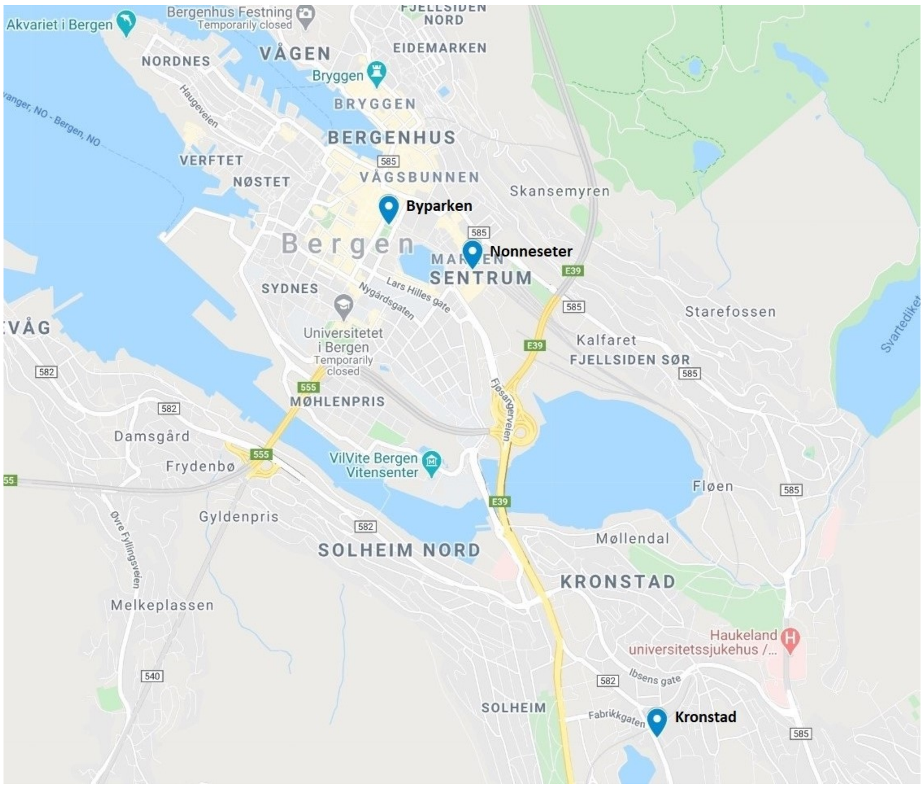


Figure 4.1: Selected locations
Credit: Google Maps (Retrieved 6/5/2020)

Chapter 5

System Architecture

In this chapter, we want to show how the planned architecture is supposed to be within our application that should be put up on MUSTs' dashboard.

5.1 The Application Architecture

We decided to use Three-tier Architecture as it is a software design pattern and a well-established software architecture. Our application architecture has three distinct layers based on the Three-tier. The architecture is built up by a client-server in which the user interface, functional process logic ("business rules"), computer data storage, and data access are developed and maintained as independent modules. We made our artifact using the rules of a three-tier architecture [55]:

1. The code for each layer must be contained with separate files which can be maintained separately.
2. Each layer may only contain code that belongs in that layer.
3. The presentation layer can only receive requests from and return responses to an outside agent.
4. The presentation layer can only send requests to and receive responses from the business layer.
5. The business layer can only receive requests from and return a response to the presentation layer.
6. The business layer can only send requests to and receive responses from the data access layer. It cannot access the database directly.
7. The data layer can only receive requests from, and return responses to, the Business layer.

8. Each layer should be unaware of the inner workings of the other layers.

By using this architecture, we have complete freedom to update and replace only a specific part in the application whenever we want instead of updating everything. This means that you can update one layer at the time without breaking the website. We can also have different teams with their areas of expertise working on the different layers to optimize efficiency and speed.

5.2 Data Layer

The data layer is as talked about based on the data lake coming from MUST. Our business layer will not have direct access to the data lake because of some potential security issues and slow speed. We opted to execute our queries in Azure Data Studio instead of in artifact and save the data in CSV-files in our development environment.

5.3 Business Layer

The business layer will in our case consist of our data filtering, model, and our Application Programming Interface (API). Our programming language of choice is Python as it is lightweight, often used when working with machine learning, and easy to import to Microsoft Azure. The business layer will read the CSV-files and apply the data filtering methods.

The layer will also be sped up by using multi-threading with a main manager. The manager will create and handle the variables used by the different processes spawned to read the data from the CSV-files. This halves the time used to read the data which results in less downtime for the API. We will also use the Norwegian Meteorological Institute API [12] together with a cache to make the calls to our API quicker.

Flask

Flask is a web framework for Python and it is classified as a micro-framework because it does not require particular tools or libraries. This means that you can get a web server running with only importing Flask and writing a few lines of code. The web server created will be used as a Representational State Transfer (REST) API where you can implement different endpoints based on HTTP methods such as *GET, POST, DELETE* [7].

Our business layer will start a Flask server after it has created and trained a machine learning algorithm. We can call our endpoint from our presentation layer and it will execute the methods in our business layer.

5.4 Presentation Layer

A user should be able to automatically get the predicted values for the rush hour the same or the next day on the main page. Page number two will consist of a form where the user can specify the input parameters to the given machine learning model and predict the number of passengers traveling from a stop with the light rail. The design should have a modern design and be easy to understand and use.

React

Using the JavaScript library React [37] means that we can create a single page application with dynamic data. A single page application is composed of individual components that can be updated/replaced independently so that the entire page does not need to be reloaded on each user action [22].

As stated earlier, the plan for our model is to have it in MUSTs' mobility dashboard. This dashboard, as of June 15. 2020, is not open to anyone other than the developers and others who are working in the MUST team. This means that no one would be able to use it right now. However, the code has been sent to them and the prediction website will be imported when they have time to implement it into the dashboard.

Chapter 6

Analysis

We will use two different types of analyses to answer our research question. The first is different evaluation metrics that will be used both when training and when testing our models. The second will be using the best of our machine learning models to show how much the rain parameter is weighted and cross-referencing that with some statistical methods.

6.1 Analysis of Rain/Ridership Relationship

We want to use the built-in methods for our machine learning models to show how much weight there is on each feature given to our models. We will focus on how much the rain parameter is weighted and its importance. The model should give us an indication of what matters when predicting the number of passengers. We will use scatter plots and statistical linear regression based on rain and passengers to check if the models are predicting the correct values.

6.2 Evaluating The Website

The website will be as basic and with as few features as possible because it will only be used to present the data given from the machine learning model. We will not create a questionnaire to evaluate the website as it will mainly consist of a single number containing a prediction. We will, however, have a constant dialogue with our partners at MUST and they will provide us feedback on what they want on the website.

Chapter 7

Design and Implementation

In this chapter, we will present how we went from raw data to a working prototype. We will also give you an understanding of why we decided to make the models the way we did.

7.1 Preparing Our Environment

The business layer is as stated made with Python together with libraries that we can import with Pip. Pip is the standard package manager for Python and is included when you install Python [36]. The first line below shows how easy it is to install a single new package. Our program does, however, need a few installs to be able to run, and we have written these into a single *requirements.txt*-file which is shown in line number two under.

```
1  pip install some-package-name
2  pip install -r /path/to/requirements.txt
```

You should be able to run the code in a new environment after a few minutes when running line number two.

7.2 Preparing The Data

As presented earlier, the data in a data lake is, more often than not, unsorted and unfiltered. Therefore, before we start making the models, we have to go through the data to make it suitable to use for machine learning. Plotting the data quickly gives us the points we have to do something about. Figure 7.1 shows the plotted data untouched. First, we have to resolve any issues inside the data. These issues can be that some of the entries inside the data set are wrongly formatted.

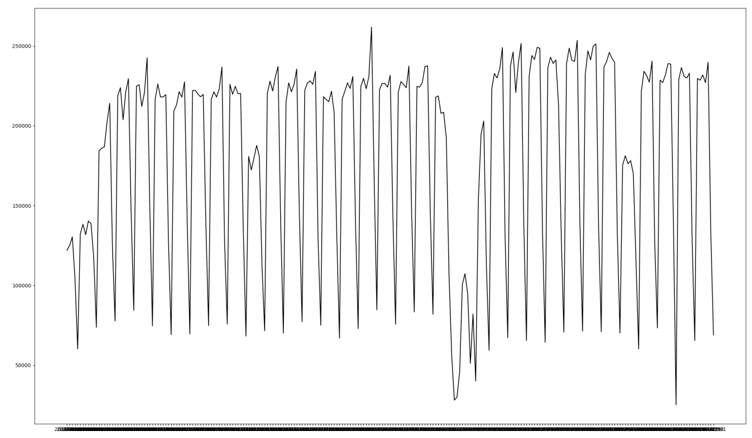


Figure 7.1: The public transport passengers per day before filtering

The second thing we could do is remove the outliers from the data or mark them so that we can filter them out later. Outliers in the data can make a huge difference when training machine learning algorithms. You can end up with a biased model if you are unlucky with the splitting of data for train and test. The outliers in our case are every time something special happens within a year. The one responsible for most of the outliers is holidays, in autumn, Christmas, Easter, and summer, and another interesting one is Black Friday. We decided to create a lookup table, containing the date and noting what type of special day it was. This means that we can train on vacation days separately if we want to. Figure 7.2 shows what we ended up with after filtering those days out.

While removing outliers tend to improve the accuracy of the regression models, it may end up removing some important information in the data that we are not aware of [2]. Bearing this in mind, we might argue that it is better to keep these in our final data for our machine learning models.

Since this is the raw data from Skyss that we got from MUST, it has hours that are missing from the dataset. The hours missing are from around 1 AM to 6 AM on Sunday through Thursday and 4 AM to 6 AM on Fridays and Saturdays. We decided it would be reasonable to try to add all the missing hours in the night so that we have data on all hours every day and night.

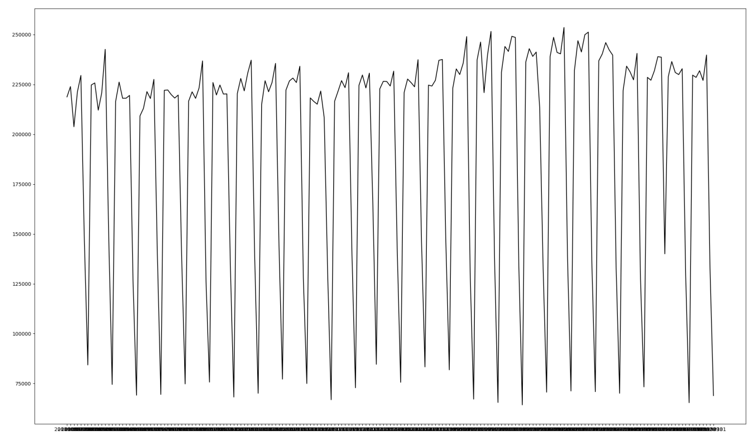


Figure 7.2: The public transport data grouped by day after filtering

Adding the missing hours may result in some other issues with the data. We might get an unbalanced dataset when adding many hours where no passengers are using public transport. Unbalanced data sets or noisy data sets appear frequently in real-world machine learning problems and increase difficulties in the training phase for models like neural networks and decision trees [20].

We had some issues picking out the parameters we wanted in our model and ultimately ended up with trying a few different configurations to give us a better end result listed below. We will also try to use *One Hot Encoder* and *Label Encoder* with different configurations to give us a better prediction.

- *Day, Hour, Rain*
- *Day, Hour, Rain, Temperature*
- *Day, Hour, Rain, Wind*
- *Day, Hour, Rain, Temperature, Wind*
- *Day, Month, Hour, Rain*

Train Test Split

We use the *train-test-split* from Scikit-learn when testing all the different configurations above. We set the *randomstate* so that we could get a consistent run every time we ran the algorithm. We specified the test size dependent on what we wanted to check. When checking the accuracy of the models, we used 70 percent as training data and 30 percent as test data.

7.3 Implementing Grid Search

Scikit-learn has a GridSearch method, however, we had a hard time using the wrapper for our models. We chose to code our own implementation for a grid search which is shown below together with the extreme gradient boost model and cross-validation.

```python
grid = {'colsample_bytree': [0.6, 0.8, 1],
        'learning_rate': [0.03, 0.05, 0.07, 0.1, 0.2],
        'max_depth': [5, 6, 7, 8, 9],
        'alpha': [0, 3, 7, 10, 15]}

for colsamp in grid["colsample_bytree"]:
    for learnRate in grid["learning_rate"]:
        for maxDepth in grid["max_depth"]:
            for alpha in grid["alpha"]:

                params = {"objective":"reg:squarederror",
                          'colsample_bytree': colsamp,
                          'learning_rate': learnRate,
                          'max_depth': maxDepth,
                          'alpha': alpha}
                cv_results = xgb.cv(..., params=params, ...)
```

7.4 Creating The Machine Learning Models

We will implement linear regression, neural network and extreme gradient boost presented in chapter 2.3 and go through the hyperparameters picked for our models.

Linear Regression

Linear regression is quick to implement and test as it has no hyperparameters. The only thing you need to do is import the model from Scikit-learn and call the fit method while passing in the trainX and trainY values.

```python
pip install scikit-learn

from sklearn.linear_model import LinearRegression

# Define the Linear Regressing model
lm = LinearRegression()

# Fitting the model to our data
```

```python
model = lm.fit(trainX, trainY)

# Predicting values
model.predict(testX)
```

Neural Network

The neural network takes much longer to configure because of hyperparameters and layers. Using grid search to try different configurations was helpful as we did not need to manually adjust the number of layers, the width of the layers, and values for the hyperparameters.

We tried a shallow network with narrow and wide layers and deep networks with narrow and wide layers. We ended up settling with a medium depth network with five wide hidden layers to get the best possible performance. The code below shows that configuration:

```python
pip install tensorflow

from keras.models import Sequential
from keras.layers import Dense

# Define the Keras model with 64-dimensional data
model = Sequential()
model.add(Dense(128, input_dim=len(X.columns), activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(1, activation='linear'))

# Fitting the model to our data
model.fit(trainX, trainY, epochs=50, batch_size=8)

# Predicting values
model.predict(testX)
```

As for hyperparameters, we tried *epochs* ranging from 10 to 100 where 10 resulted in an under-fitted model and 100 in an over-fitted model. 50 ended up being a good result in collaboration with the other hyperparameters. Changing from *Relu* and *Linear* as *activation function* resulted in almost halving the accuracy. *Batch_size* got tuned dependent on the speed of the computer and checking the accuracy and ended with a value of 8.

There were a few hyperparameters we ended up not using because of different reasons. The base *learning rate* worked the best which is why we ended up removing it. *Dropout* resulted in under-learning the network even when using a reasonably low value. The same happened with *momentum* only it overshot the minimum and resulted in worse accuracy.

Extreme Gradient Boost

The extreme gradient boost model takes a short amount of time to configure if we are only using the base values, however, there are many hyperparameters as presented earlier. The code under shows the simplest use of the XGBoost library if you have well-structured data. Figure 7.3 under the code visualize how the tree could be structured after it has run.

```
pip install xgboost

import xgboost as xgb

# Importing the model
xgb = xgb.XGBRegressor()

# Fitting the model to our data
xgb.fit(X, y)

# Predicting values
xgb.predict(testX)
```

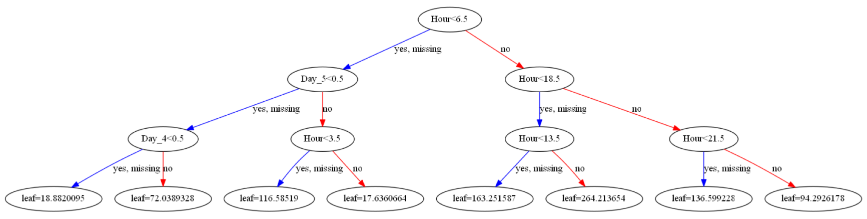


Figure 7.3: A visualization of a tree

Testing *learning rate* with values ranging from 0.03 to 0.3 would help us reach the *max depth* going from 4 to 10. The final values for *learning rate* ended up being 0.1 together with a *max depth* of 8. We picked 1000 for our *num_boost_rounds* together with a *early_stopping_rounds* value of 10. This means that the model will be able to learn many times if needed with short trees or it will hit the early stopping if the accuracy is not increased after 10 trees.

To generalize the model, we tested *colsample_bytree* with values from 0.5 to 1. 0.5 means that it will take half of the columns give to the model and 1 means that it will take everything. We also tested values for *alpha* ranging from 0 to 20. The accuracy ended up best when only using one of the generalization methods. This means that the *colsample_bytree* got the value 1 to enable all columns and *alpha* ended up as 15.

For the cross-validation, we sat a static number as our *seed* so that every round of cross-validation would do the same splitting of the data to get less variation in the accuracy. We also sat the cross-validation to 5-fold meaning it would use 20 percent of the training data as testing for each round.

Chapter 8

Results and Interpretation

One of the steps in our research methodology design science talked about in Chapter 3 is to evaluate the artifact. In this chapter, we want to evaluate our model and present our findings.

Given the data and manually testing our models with different input parameters, we ended up not using LabelEncoder or OneHotEncoder as these did not give us a higher accuracy when training on the input parameters $[Day, Month, Hour, Rain]$. If we were to use the labeling of the type of special day, such as autumn vacation or summer vacation, we would need to use both of them to get to the same prediction accuracy as when only using $[Day, Month, Hour, Rain]$. The model would be more complex when adding the extra parameter and the manual prediction would be harder to do. We also did not add the missing hours as this made our dataset unbalanced and gave a worse prediction.

The rest of the evaluation of the machine learning models will be done with manual prediction where the day, month, and hour will stay the same for every prediction while rain will change for every prediction. The manual testing will be done on a preset of downfall defined from 0 to 10 millimeters and we will do it with a step-size of 0.1. The table 8.1 shows a section as an example taken from a prediction made with the neural network:

$$\begin{array}{c} \begin{array}{cccc} \text{Day} & \text{Month} & \text{Hour} & \text{Rain} \end{array} \\ \begin{bmatrix} 0 & 2 & 8 & 0 \\ 0 & 2 & 8 & 0.1 \\ 0 & 2 & 8 & 0.2 \\ 0 & 2 & 8 & 0.3 \end{bmatrix} \end{array} = \begin{array}{c} \text{Predicted} \\ \begin{bmatrix} 876 \\ 868 \\ 861 \\ 854 \end{bmatrix} \end{array}$$

Table 8.1: Showing a section of the manually predicted values from Byparken Monday February 8:00 with a Neural Network

8.1 Evaluating our models

Shown in Chapter 7, Design and Implementation, we made three models. We have to compare the results from our machine learning models and compare their strengths and weaknesses to be able to choose the better model between the two. We will also observe how they perform by using the evaluation metric Mean Absolute Percentage Error, called *accuracy* going forward.

It was easy to remove the simplest model, linear regression, as this resulted in a much worse score compared to the two others at around 60 percent accuracy. We suspected that the linear regression model would be worse, as it just uses math to tweak its parameters. As shown in figure 8.1, it has a hard time with the points outside a number from the medium. It is fast to learn and predict, but that would not be good enough for us since the neural network and extreme gradient boost has much better accuracy.

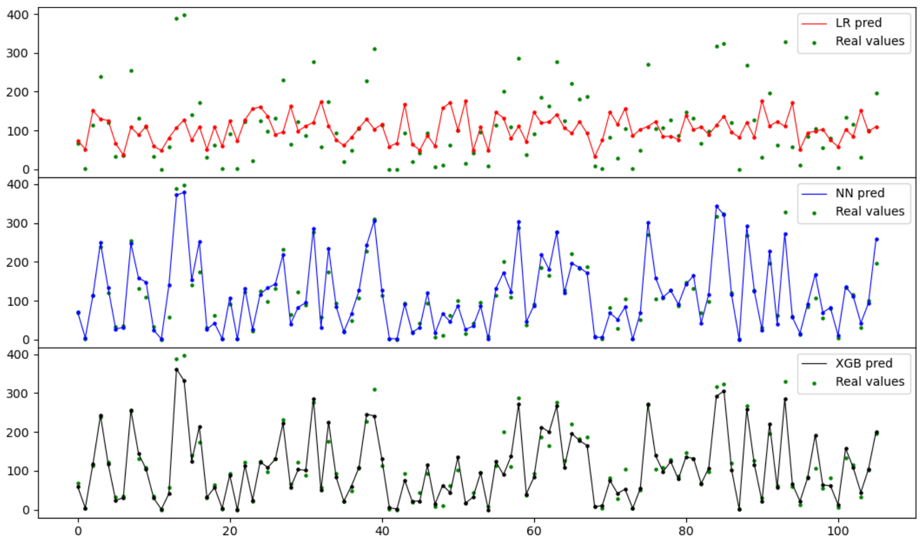


Figure 8.1: Predicted passengers Monday to Friday in May 8:00 to 9:00

The neural network and the extreme gradient boost were, however, a little harder to evaluate as they almost got the same accuracy. The neural network got around 83-88 percent accuracy and the extreme gradient boost got 85-90 percent dependent on which stop we trained our model on. The neural network seems to be more dependent on the amount of rain in an hour compared to XGBoost. This can be seen in figure 8.2 where the predicted number of passengers goes down when there is more rain. The evaluation would then need to be backed by some other data as seen in figure 8.3. Looking at that plot shows that the number of passengers increases slightly with the amount of rain. The blank spots in the figure are where we don't have any data.

The neural network is not performing as well as the accuracy tells us when we are comparing figure 8.2 and figure 8.3. Figure 8.2 shows a decreasing number of passengers and figure 8.3 shows an increasing number of passengers on average compared to no rain at all. Most other predictions from the neural network show the same tendency with a much lower predicted number when there is a lot of rain. Our contact from Skyss also told us that the extreme gradient boost is more in line with their internal findings. This means that the neural networks result is not being generalized enough or the neural network is not able to handle the unbalanced data in the same way as the extreme gradient boosting model.

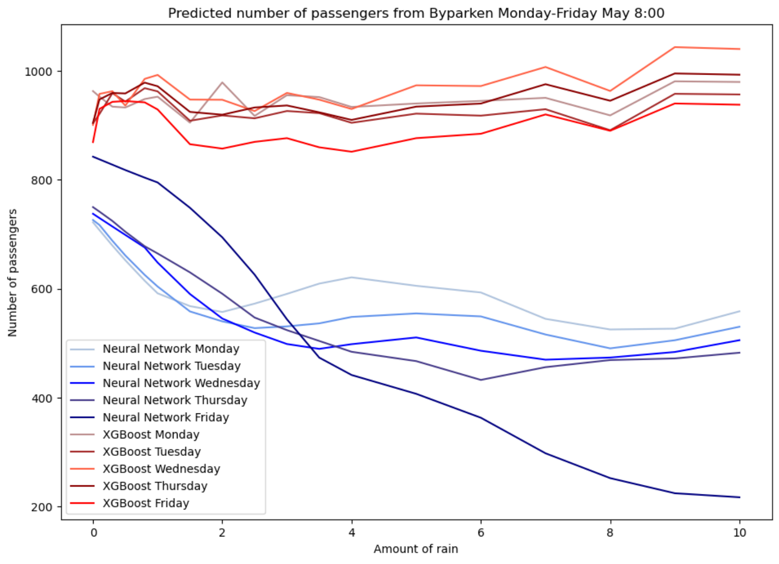


Figure 8.2: Predicted passengers Monday to Friday in May 8:00 to 9:00

We will, therefore, use the extreme gradient boosting model for the rest of the results presented within this thesis. The model has an error margin of 10 to 15 percent wrong dependent on some random variables and which stop we train and validate on. Besides, it is quick to learn which is a plus for when MUST has downtime on the website when training a new model based on newer data.

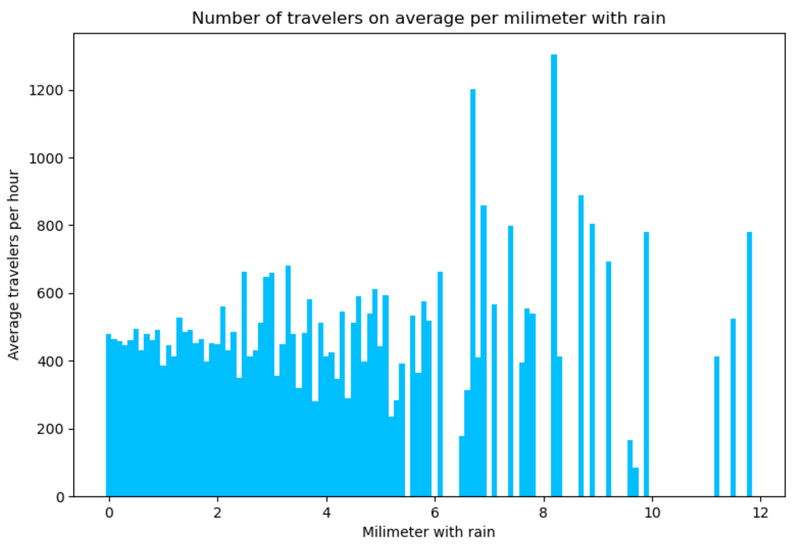


Figure 8.3: Average number of all passengers grouped by the amount of rain traveling from Byparken

8.2 Rain and Ridership Relationship

We have seen earlier that the number of passengers is dependent on how much rain there is, but we have not addressed how much it changes. Using the extreme gradient boost model, we can plot the importance of each feature used in the model using two different types of feature importance. *Gain* which is the average gain across all splits the feature is used in and *Weight* which is the number of times a feature is used to split the data across all trees [53]. We will now present our findings on the different stops described in section 4.3.

Byparken

Byparken has the highest number of passengers in an hour and is the stop we have used the most to validate our models. Figure 8.4 shows the feature importance for gain on the left and weight on the right. This tells us that the rain does not affect the number of passengers as much as hour and day, but it has many splits meaning that for fine-tuning of the prediction it uses the weather and therefore it has some significance. That significance is also backed up by a regression line shown in figure 8.5.

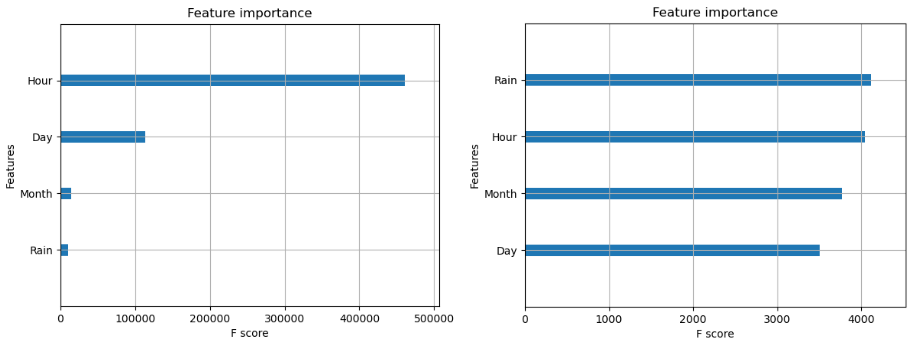


Figure 8.4: The feature importance gain and weight for Byparken

Figure 8.5 plots all the data separated by Monday to Friday and Saturday to Sunday without looking at the month, hour, or filtering out the days where something special happened. This means that there might be a bias in some of the points and they could be too high or low dependent on what happened. We do however see that the model predicts right when predicting a small increase when there are small amounts of rain. Looking at the heat map displaying the correlation coefficient for all the features used in the machine learning model in figure 8.6, we can see that rain has little effect on what the number of passengers should be.

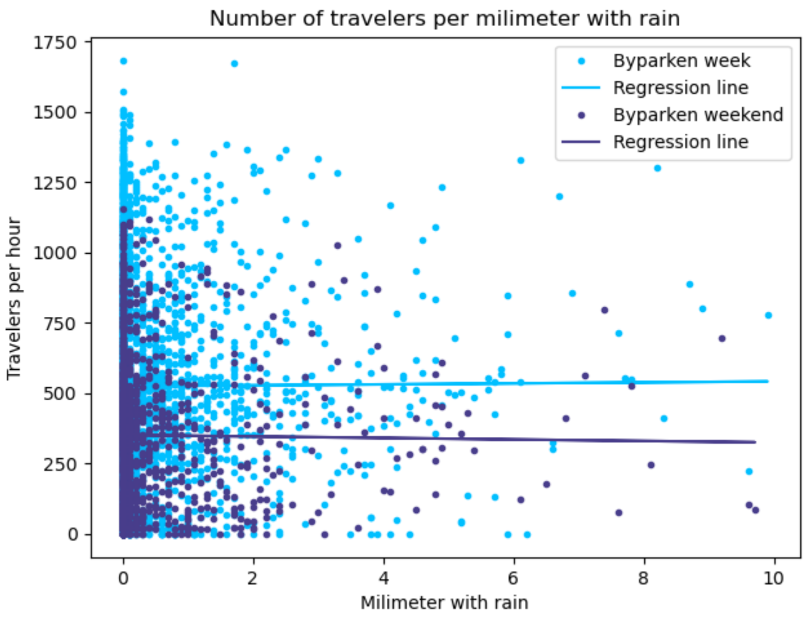


Figure 8.5: Regression of passengers traveling based on rain from Byparken

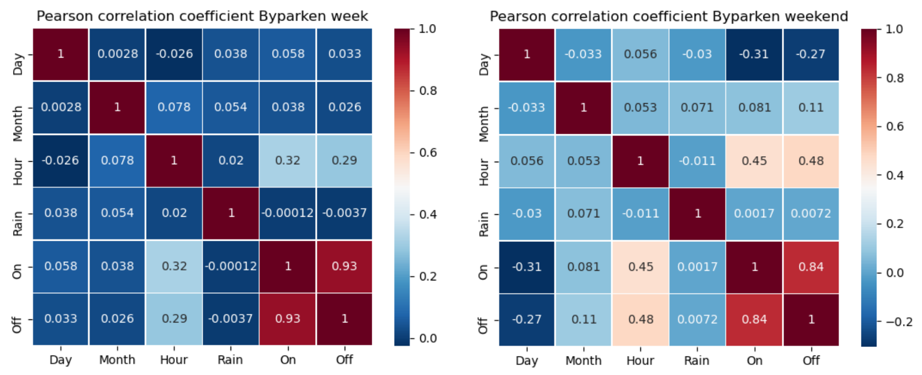


Figure 8.6: Pearson correlation coefficient Byparken week and weekend

Comparing figure 8.7 and figure 8.8 indicates that the rain increases the number of passengers greatly independently of if it is rush hour or a normal hour mid-day. Rain on Sunday at 13:00 makes the number of passengers go down for the most part, however with Monday to Saturday, the rain makes the number of passengers go up or stay almost equal as best seen in figure 8.7. Looking at the rush hour at 8:00, we can see that the model predicts an increase in ridership on most days. Monday seems to be down for all amounts of rain and Tuesday to Friday seems to have a decrease when there are approximately 4.7 millimeters of rain.

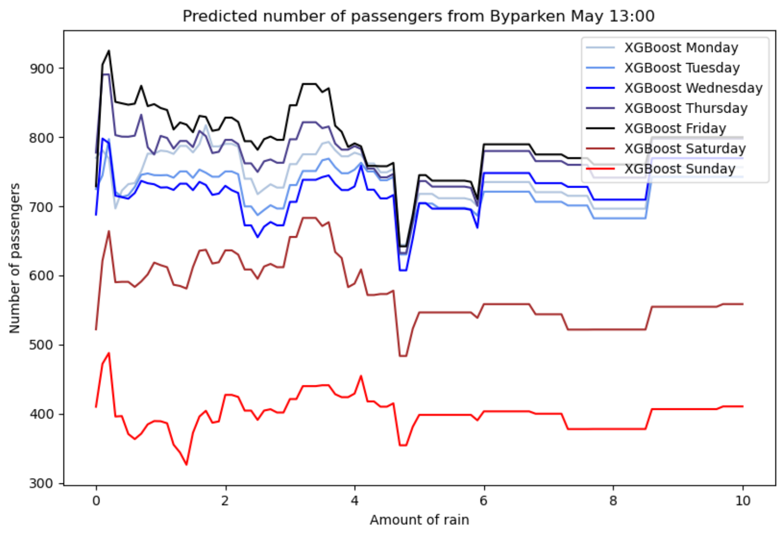


Figure 8.7: Number of passengers traveling from Byparken based on rain 13:00

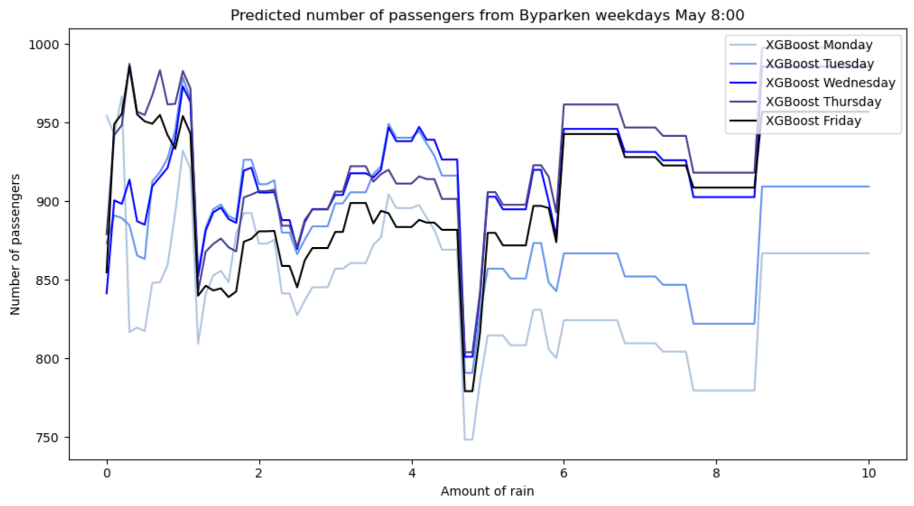


Figure 8.8: Number of passengers traveling from Byparken based on rain 8:00

Nonneseter

The feature importance for the model is the same for Nonneseter as for Byparken (figure A.5). Comparing the regression lines from Byparken (figure 8.5) and Nonneseter (figure 8.9) shows a slight increase in passengers travelling from Nonneseter when there is more rain. We can also see that the correlation is higher in Nonneseters' correlation coefficient matrix shown in figure 8.10 compared to Byparkens' in figure 8.6.

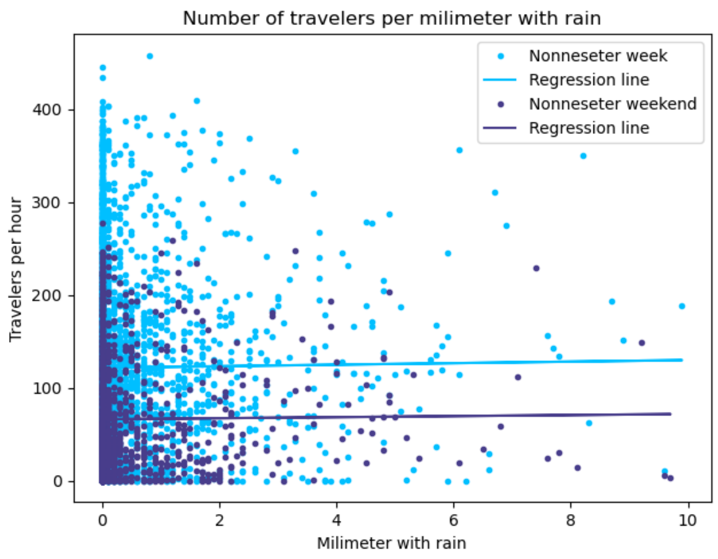


Figure 8.9: Regression of passengers traveling based on rain from Nonneseter

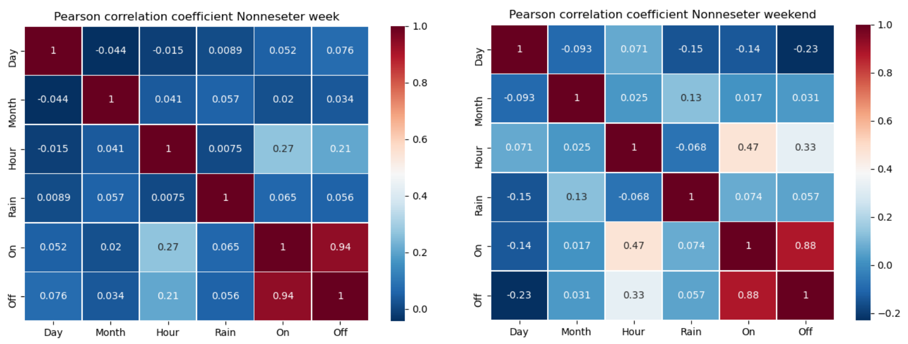


Figure 8.10: Pearson correlation coefficient Nonneseter week and weekend

Looking at figure 8.11, the figure shows a spike in ridership on most days at 13:00 when there are small amounts of rain. The spike does not last long on Saturday and Sunday, but on Friday it lasts to about 4 millimeters of rain. Looking at the rush hour in figure 8.12, we can see the same spike lasting only to approximately 1.8 millimeters of rain before it flattens out. Monday seems to have the same decrease as Byparken meaning that we might have a special day where fewer citizens used public transport.

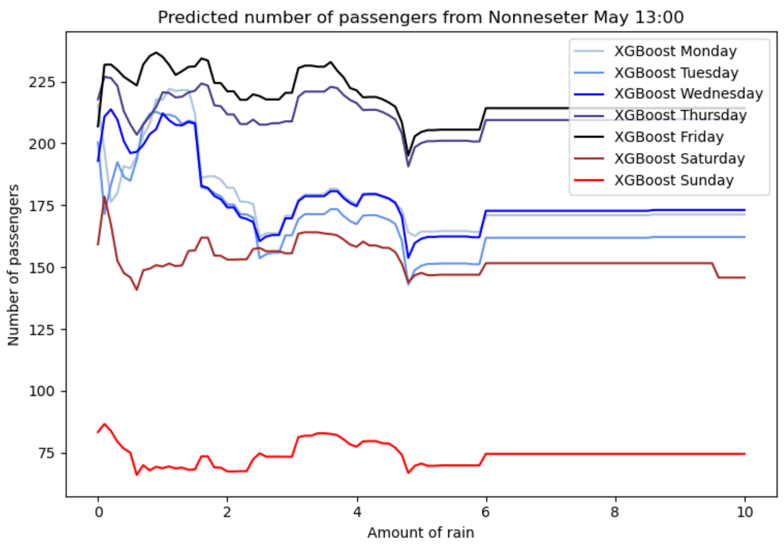


Figure 8.11: Number of passengers traveling from Nonneseter based on rain May 13:00

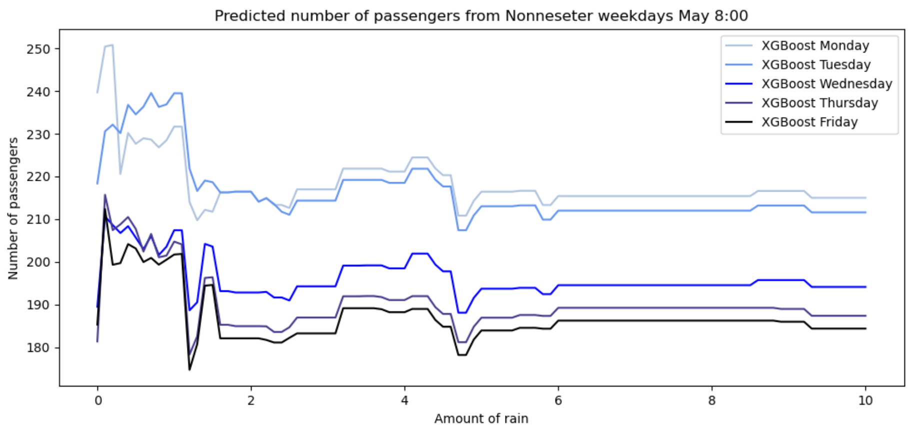


Figure 8.12: Number of passengers traveling from Nonneseter based on rain May 8:00

Kronstad

Figure 8.13 shows a regression line for Kronstad pointing a little more down compared to Byparken and Nonneseter. This means that rain has a slightly bigger negative effect on Kronstad. This is not reflected in the correlation matrix 8.14 as it has almost the same values as Nonneseter when looking at the *On/Rain* cell.

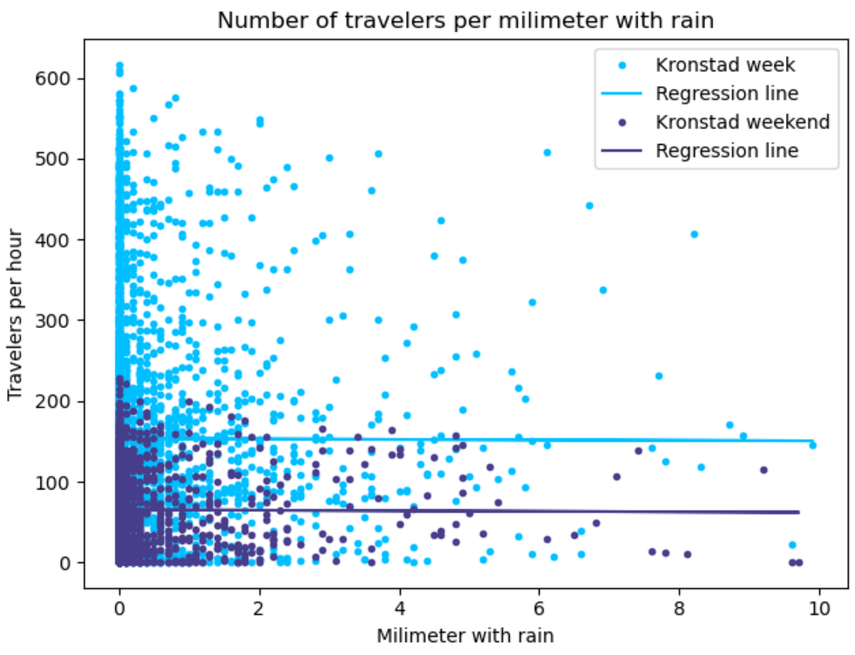


Figure 8.13: Regression of passengers traveling based on rain from Kronstad

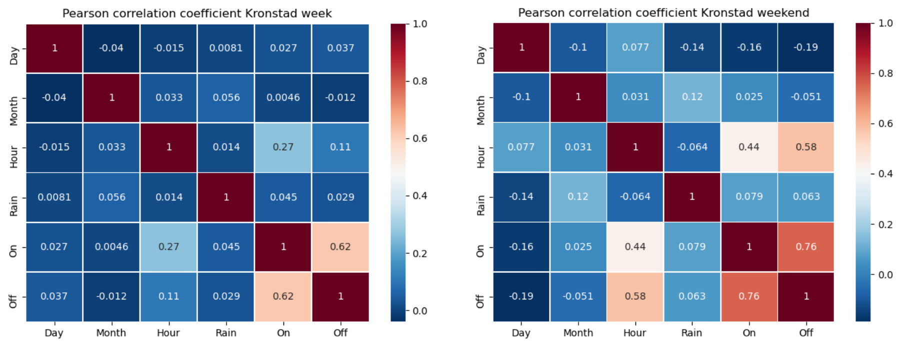


Figure 8.14: Pearson correlation coefficient Kronstad week and weekend

The predicted values from our machine learning model seen in figure 8.15 do however give us a more consistent answer for each day compared to Byparken and Nonneseter. We see that every weekday at 13:00 have a passenger increase when rain is between 0.1 and 1.8 before going inconsistently down until there are 4 millimeters of rain. We also see that rain only has a small effect on the number of passengers during the weekend. Figure 8.16 with predictions from 8:00 only shows a decrease most of the time when there is rain. Monday seems to have a large spike early and Friday has a large dip around 1.8 millimeters.

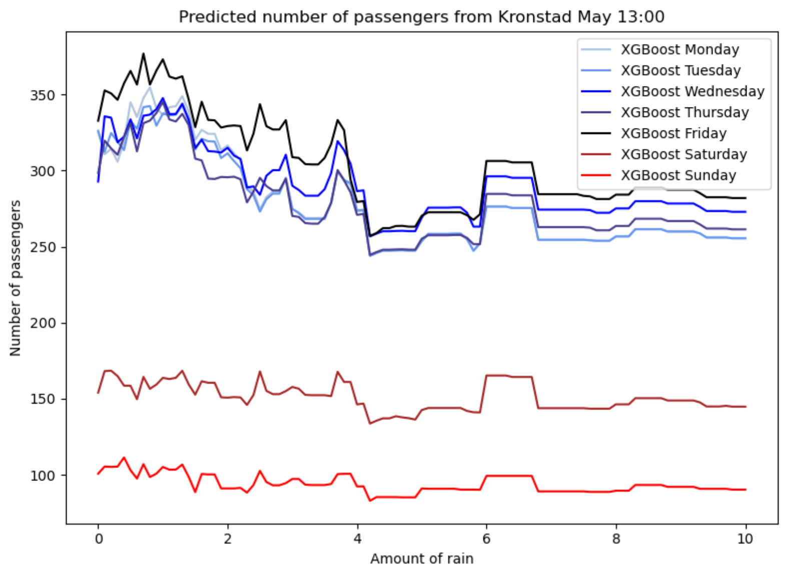


Figure 8.15: Number of passengers traveling from Kronstad based on rain May 13:00

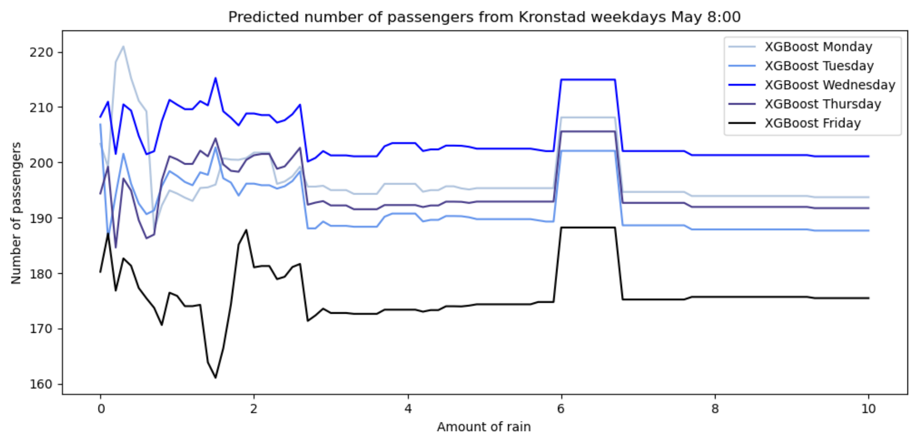


Figure 8.16: Number of passengers traveling from Kronstad based on rain May 8:00

8.2.1 Summer Vacation

As stated in Chapter 1.2, we also wanted to look at how the summer month with a public holiday compared to the rest of the year. The public holiday in Norway is in July and so we plotted the average value for the whole month to check for any similar patterns between July and May or November. Figure 8.17, 8.18 and 8.19 shows these predicted values by percentage.

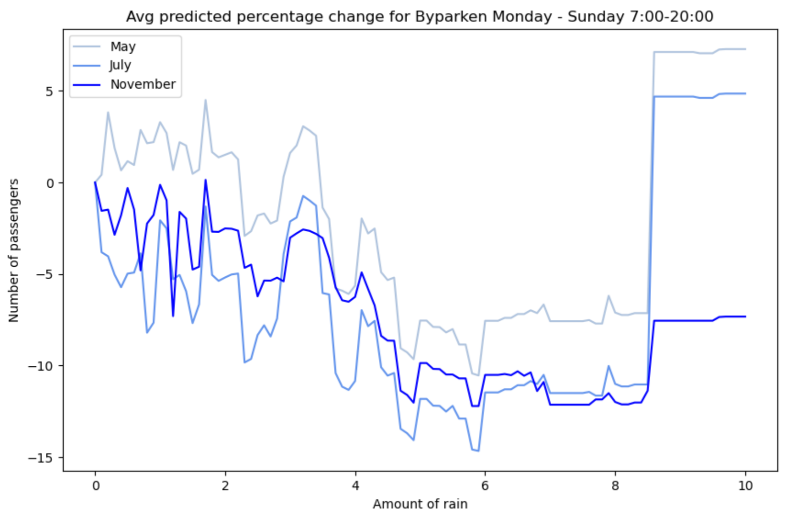


Figure 8.17: Number of passengers from Byparken grouped by month including all days 7:00-20:00

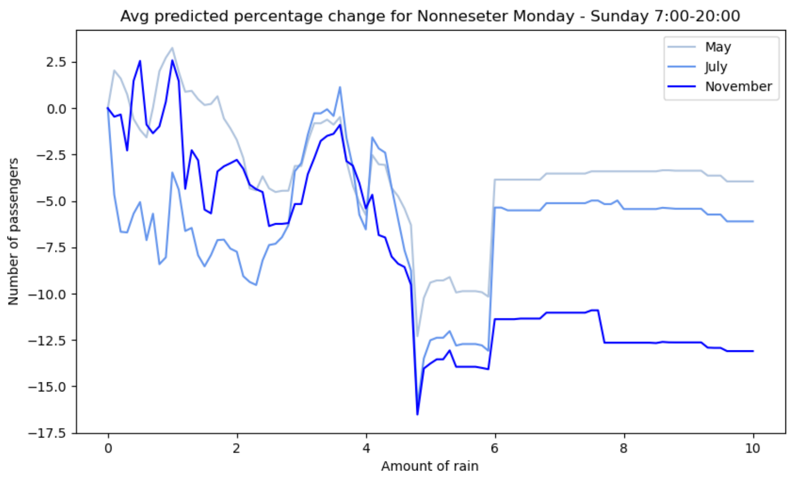


Figure 8.18: Number of passengers from Nonneseter grouped by month including all days 7:00-20:00

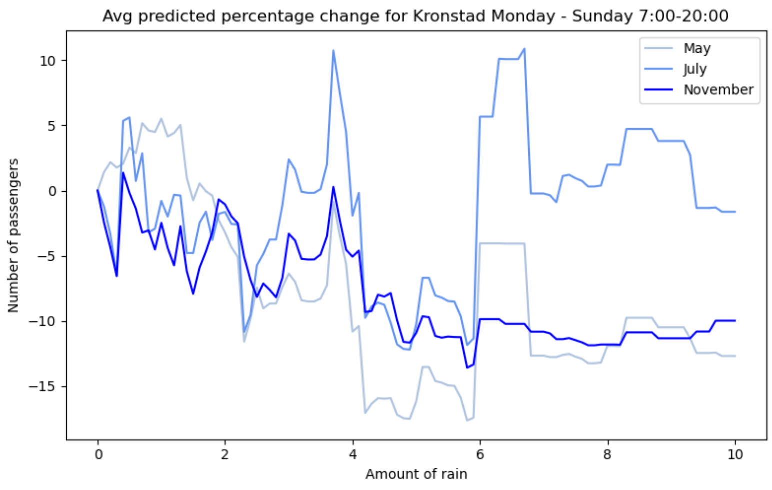


Figure 8.19: Number of passengers from Kronstad grouped by month including all days 7:00-20:00

Figure 8.17, 8.18 and 8.19 show that May always have an increase when there are small amounts of rain on all the stops and the increase changes to a decrease around 2 millimeters. November has a decrease at both Byparken and Kronstad, but has a spike on Nonneseter. We can see that rain in July has a substantial decrease in ridership on Byparken and Nonneseter. July at Kronstad has a quick spike down before going up again.

8.3 Summary

Pearson correlation coefficient matrix and linear regression are not giving us a specific enough answer when working on the whole dataset. The linear regression is almost flat on all the stops, both on *On* and *Off*. The coefficient matrix shows a low correlation between *Rain* and *On* or *Off*. This shows that the rain itself does not have a big impact compared to *Hour*, *Day* and *Month*, however, our machine learning model shows that there is a correlation. This means that the rain affects the number, but only for more fine-tuning after all the other parameters and gives up to 15 percent change in passenger numbers.

As for the model itself, it seems to give a good prediction with all the average values going down until 5.5-6 millimeters which seem to be the diminishing return for our model. The sudden spike followed by a flat line is showing up somewhere around 6 millimeters means that we do not have enough data for higher amounts of rainfall and that the model is underfitted. The predicted number of passengers traveling when there is a higher amount of rain is more dependent on what day and hour it is or if something special happened that day. The model would then be biased with a higher or lower prediction for that exact amount of rain and the flat line shows that the model has opted to generalize or there are missing data. On the flip side, we can see that the model is not over-sensitive to special days because of the regularization.

There are a few possibilities as to why the result for rain and ridership is as presented when we have less than 6 millimeters of rain. We have at least three groups of citizens using public transport in different ways. Group 1 always uses public transport and does not care what the weather is, group 2 decides to stay home or use a car instead of taking public transport dependent on the weather, and group 3 decides to use public transport instead of walking or biking dependent on the weather. This means that the passenger numbers can go up or down dependent on how big those groups are.

Many of the predicted graphs have the same tendency with a spike early, slowly going down to around minus 10 percent around 6 millimeters and followed with a new spike. We estimate that many citizens are walking in the city center when there are 0 millimeters of rain and get on public transport when the rain starts giving us the spikes from 0.1 to 1.5 millimeters. Citizens are used to small amounts of rain, but with lots of rain, they might stay home or use their cars for convenience resulting in a decrease in ridership.

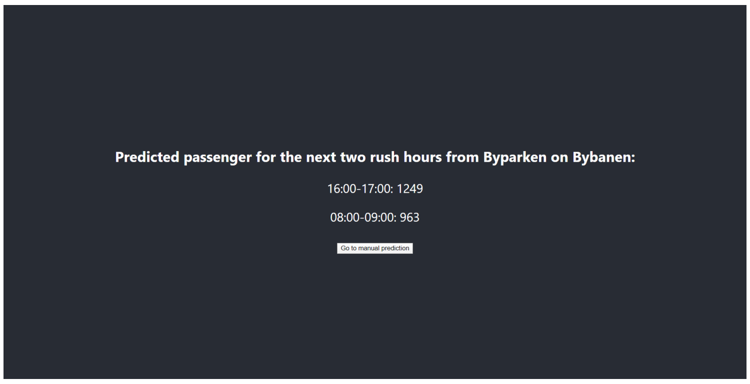
Chapter 9

Web Application and Survey

This chapter contains what we have done for the citizens and what the citizens have done for us, the web application and survey respectably. The web application was made so that the citizens use the artifact themselves and the survey was made to check what the citizens thought about public transport.

9.1 Web Application

As described in step 4 in our research methodology, we will now exhibit our web application used for predicting travelers. The user is presented with the main page containing the next two rush hours, 8:00 to 9:00, and 16:00 to 17:00 sorted by which comes first. The time has a corresponding prediction based on the weather forecast for tomorrow automatically found by our model using the Yr API. The design is presented in figure 9.1.



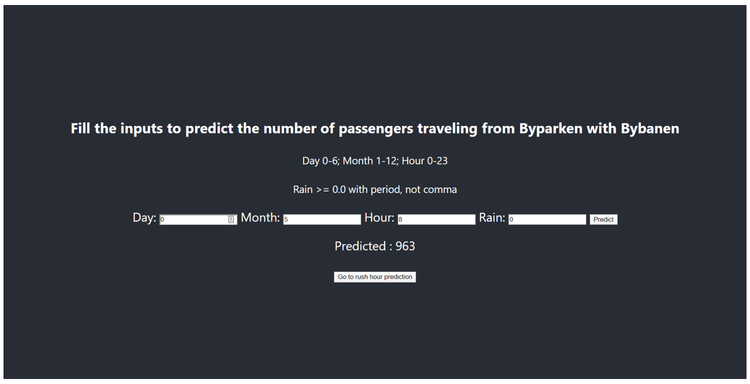
Predicted passenger for the next two rush hours from Byparken on Bybanen:

16:00-17:00: 1249

08:00-09:00: 963

Figure 9.1: Homepage on website

Clicking the button in at the bottom in figure 9.1 changes the render to the manual predicted web page shown in figure 9.2. The user can write a specific day, month, and hour and test with different amounts of rain based on the instructions given on the page. This page is useful for this thesis and the others from MUST to show how the number of travelers is dependent on how much rain there is. This page is, however, not meant to be adapted into the MUST dashboard, but they can provide a link so that the citizens in Bergen could go on this page if they are interested.


Fill the inputs to predict the number of passengers traveling from Byparken with Bybanen

Day 0-6; Month 1-12; Hour 0-23

Rain >= 0.0 with period, not comma

Day: 0 Month: 5 Hour: 8 Rain: 0

Predicted : 963

Figure 9.2: Manual prediction on website

9.2 Survey

As presented earlier, we also created a self-administered questionnaire to check if the consensus is the same as our findings in the data. The rest of the survey not presented here will be in the appendix. When making the questionnaire, it was important to make it using a set of rules as much as possible [33]. These rules are:

1. Keep the questionnaire as short as possible.
2. Ask short, simple, and clearly worded questions
3. Start with demographic questions to help respondents get started comfortably.
4. Use dichotomous (yes or no) and multiple-choice questions
5. Use open-ended questions cautiously.
6. Avoid using leading questions
7. Pretest a questionnaire on a small number of people.

8. Think about the way you intend to use the collected data when preparing the questionnaire.

104 citizens answered our survey and there were half and half with students and working citizens, see 9.3 on the left side. We can also on the right side in figure 9.3 see that persons answering the survey live relatively close to the city center and 89 percent uses 0 or 1 bus/light rail/boat to get to Festplassen and 54 percent uses 10 minutes or less to get there, 9.4.

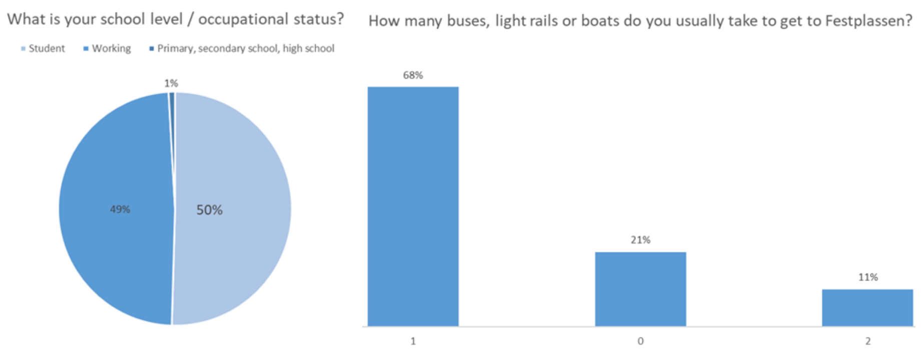


Figure 9.3: Occupational status and number of buses

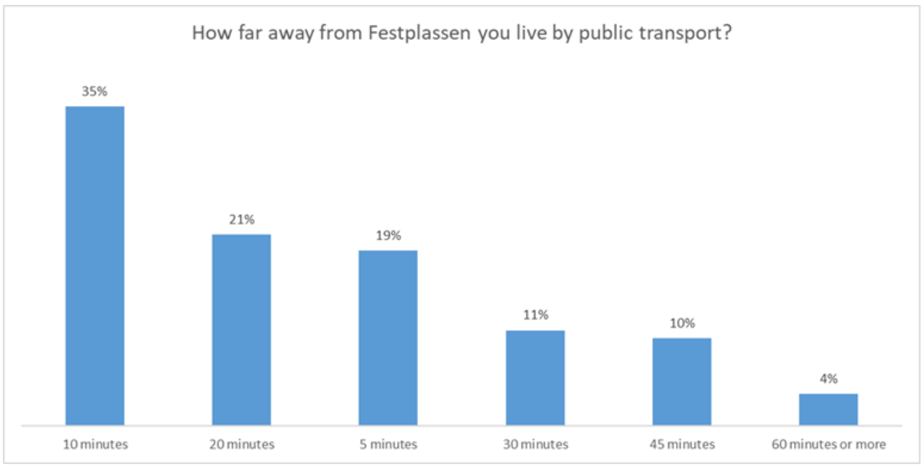


Figure 9.4: Traveltime to get to Festplassen

The survey also shows that 66 and 73 percent of the users do not care if there is rain when they have scheduled plans, alone or with friends/family respectively. 26 percent of citizens are staying home and 8 percent decided to use a car instead of public transport when they had plans alone as seen in figure 9.6. The number of citizens staying home when they have plans with friends or family is 13 percent and 13 percent will use a car instead as seen in figure 9.5.

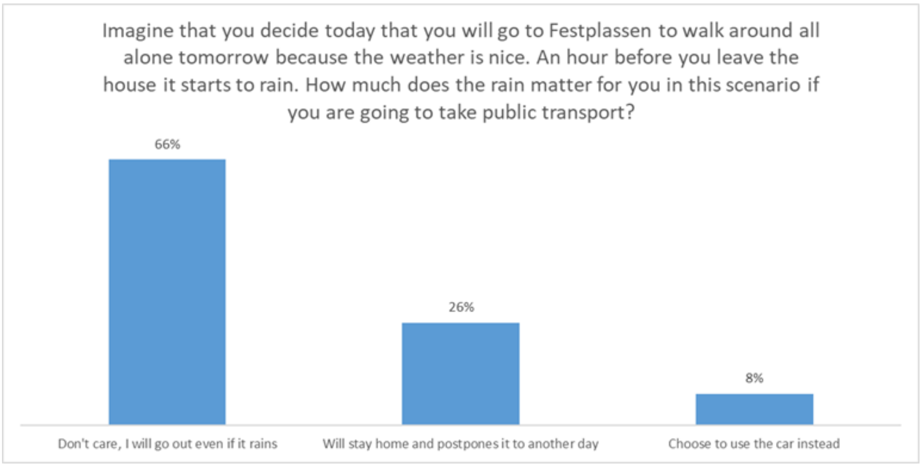


Figure 9.5: Does rain matter traveling alone

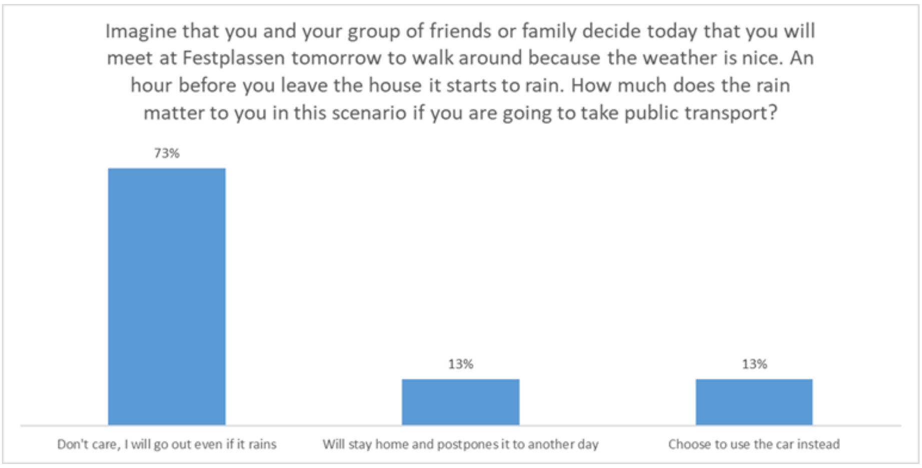


Figure 9.6: Does rain matter traveling to meet friends

In more general terms, 65 percent of users say that they do not care about the rain when they are going to use public transport and 32 percent say that they care, seen in 9.7. Comparing this to how our machine learning model predicts, we can see that 32 percent is much higher than the minus 15-16 percent we see in our average percentage plots in figures 8.17, 8.18 and 8.19. The question is quite general and there are lots of cases to think about. The users might think about heavy rainfall or if they were going to take public transport without any reason.

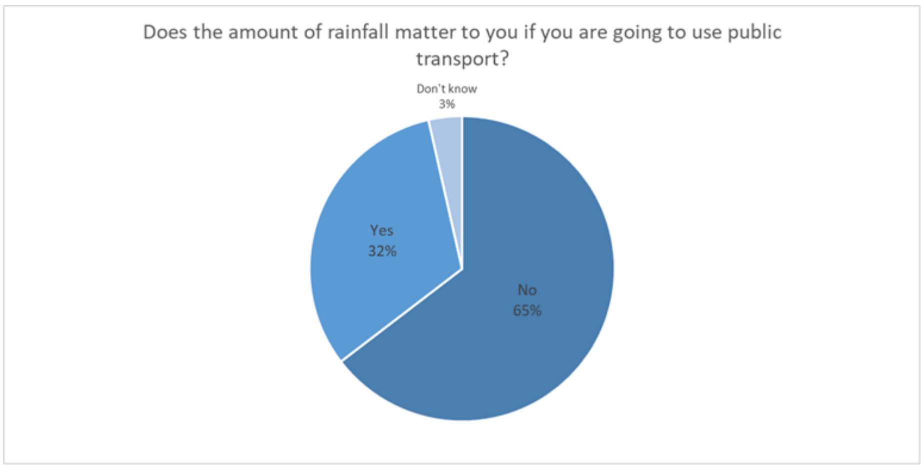


Figure 9.7: Amount of rain affect public transport

Chapter 10

Threat To Validity

The results of a research project will always be accompanied by certain sources of error, coming from both the planning and processing of the data and creating an artifact. We will present some information based on the threat to validity and what we did to counteract it.

Inadequate preoperational explication of constructs means that the constructs are not sufficiently defined before they are translated into measures or treatments. This includes when two inspection methods are compared and it is not clearly enough stated what being 'better' means [50]. We have tried to eliminate this by defining what we want to measure when looking at the rain/passenger relationship and explaining what we think is better in the context of machine learning in terms of both the model itself and the accuracy.

Using a single type of measures or observations involves a risk that if this measure or observation gives a measurement bias, then the experiment will be misleading and we would have *mono-method bias*. By involving different types of measures and observations they can be cross-checked against each other [50]. The machine learning model made is based on three locations and different days, and the statistics and survey should help to eliminate the *mono-method bias*.

The survey is subject to social threats where they may, based on the fact that they are part of an experiment, act differently than they do otherwise, which gives false results from the experiment. *Hypothesis guessing* and *experimenter expectancies* talk about how the citizens answering the survey might try to figure out what the purpose and intended result of the experiment. This includes unconsciously or consciously changing their answers, positively or negatively, based on what they expect from the experiment. One other issue here is *instrumentation* meaning that the experiment could be affected negatively if data collection forms are badly designed [50].

We have also tried to make the machine learning code more generic to be able to run it on other datasets, meaning that we can use it on the bike or car data MUST have in the data lake helping with the *restricted generalizability across constructs* [50]. To do so, the models have to be specifically made for the public transport data first and then rewritten to be able to adapt to different types and sizes of data-sets. After this has been done it has to be retested on the public transport data to validate that the accuracy and generalization worked as it did before.

On a more general note, we have always been two students and two student advisors evaluating between all the different steps in the development process with bi-weekly meetings. Some of these meetings have also been with developers and researchers from MUST and Skyss, and ended up with us getting a machine learning model and research that could be used in the real world.

Chapter 11

Conclusion

We will in this chapter use our findings and summarize how the findings resulted in regards to our sub-goals and research questions presented in section 1.2. We will then give a recommendation about how Skyss and other transport companies can use our research, and end with some thoughts we have for future work and projects.

Sub-Goals

To create a website that MUST could use for front-end and a better model for machine learning in the backend, we defined two sub-goals. We will present the conclusion to them here using the results from the earlier chapters:

- *"Use different machine learning models to find the best one for our data."*
  Our work has shown in section 8.1, that this was important to do as the result for each model was different from each other. The multiple linear regression did not work well at all. The neural network and extreme gradient descent were closer to each other, but we chose the extreme gradient descent model as it came out ahead with higher accuracy and fit for our problem.
- *"Create an easy to understand website that can be ported to MUST."*
  The website we made was as simple as possible while still giving the information needed for our use-case. The basic front page will be the one implemented on MUSTs dashboard at a later date, and the manual prediction page was made to easier showcase the effect rain had on passengers on public transport.

Research Questions

Through this thesis, we created an artifact containing machine learning and a website using design science research. The artifact was made in collaboration with MUST and working together we researched what was best for their dashboard and how to get better predictions. By using their data, we have successfully created an easy to use implementation with both a backend and frontend, and we can answer the research questions:

- *"How good of a prediction of ridership on Bergen light rail can we get with machine learning?"*
  Our results show that the prediction does well in predicting the number of passengers using the light rail. There are a few issues when predicting when there are more than 5.5-6 millimeters with rain because we do not have as much data for these numbers to give an accurate prediction. It does, however, seem to not be over-sensitive to special days because of the regularization.

- *"How much correlation is there between rain and the number of passengers on public transport?"*
  In this report, we have shown that the correlation seems to be present at every stop, but it is hard to find the correlation in the Pearson correlation coefficient matrix as the rain correlation is much smaller than the other labels. Looking at the linear regression also gave almost a stagnant regression line. The correlation shows up after *Hour*, *Day* and *Month* and is used for fine-tuning instead of showing accurately what the number of passengers should be. The machine learning shows up to a 16 percent decrease when there is rain.

- *"Does rain in the summer have a higher impact on passenger numbers on public transport compared to the rest of the year?"*
  Our work has shown rain has in most cases a higher impact on the number of passengers riding public transport. The presented results earlier showed that the average number of passengers for a whole month went more down in the summer compared to May and November.

11.1 Recommendation to Skyss

Our research shows that the machine learning model works well from a technical standpoint. We want to recommend Skyss to start their machine learning project that they can use in their company to be able to predict the number of passengers on each stop and every line based on the weather. The public transport could be dynamic and dependent on how the weather would be the next hours or the next day. Skyss could end up saving money by using fewer busses and workers at different times throughout the day. There would be some obstacles, however, they have some time to figure that out before the citizens would be ready for a dynamic public transport offer instead of the static we have today.

Skyss has an app, but it is not used for anything that can analyze the travel habits of its inhabitants. There have been many times where we have wished that Skyss tracked every traveler when they used the app when using public transport. There are some privacy concerns here, but the amount of data coming from an app like that would make us much better understand the travel habits.

11.2 Future Work

There were many thoughts in the air after all those meetings we have had. There are more stops to analyze to check if there is a difference between the accuracy of the light rail and the bus. There could also be research about how rain affects the number of travelers from the countryside. Oslo has, as talked about in chapter 1, just over half the number of rainy days as Bergen. We can get more information on how the weather affects those living in a rainy or sunny city if someone would do the same research in Oslo.

MUST has data about most of the transportation in Bergen, that is why an analysis based on how the weather affects the correlation between different types of transport might be helpful to better understand how citizens move in a city. To further understand the use of personal vehicles, one could look at the correlation between rain, car, and public transport. Someone could even do the same research as us, but add wind, temperature, and humidity to see how that affects the number of public transport users.

Getting access to data based on the location from an app or through a telecommunications company and analyzing it instead would greatly increase the understanding of how citizens move on a lower level. We could better understand where citizens are coming from and going to and how far citizens are willing to travel when there is sun or rain.

Bibliography

[1] Hans Asbjørn Aaheim and Karen Evelyn Hauge. “Impacts of climate change on travel habits. A national assessment based on individual choices”. English. In: (2005).

[2] Guilherme A Barreto and Ana Luiza B.P Barros. “A Robust Extreme Learning Machine for pattern classification with outliers”. eng. In: *Neurocomputing* 176 (2016), pp. 3–13. ISSN: 0925-2312.

[3] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization”. In: *Journal of machine learning research* 13.Feb (2012), pp. 281–305.

[4] Bouvet. *About Bouvet.* 2019. URL: `https://en.bouvet.no/about-bouvet` (visited on 02/25/2020).

[5] Vanessa S Brum-Bastos, Jed A Long, and Urška Demšar. “Weather effects on human mobility: a study using multi-channel sequence analysis”. eng. In: *Computers, Environment and Urban Systems* 71 (2018), pp. 131–152. ISSN: 0198-9715.

[6] Hatem A Fayed and Amir F Atiya. “Speed up grid-search for parameter selection of support vector machines”. eng. In: *Applied Soft Computing Journal* 80 (2019), pp. 202–210. ISSN: 1568-4946.

[7] Flask. *Quickstart.* 2020. URL: `https://flask.palletsprojects.com/en/1.1.x/quickstart/` (visited on 04/25/2020).

[8] Vestland fylkeskommune. *Ny rekord i kollektivreisande.* 2020. URL: `https://www.vestlandfylke.no/nyheitsarkiv/2020/ny-rekord-i-kollektivreisande/` (visited on 03/19/2020).

[9] Daniel Graupe. *Principles of artificial neural networks.* eng. Singapore ; 2013.

[10] Ar Hevner et al. “Design science in Information Systems research”. English. In: *Mis Quarterly* 28.1 (2004), pp. 75–105. ISSN: 0276-7783.

[11] Susan Holmes. *Stanford - RMS Error.* 2000. URL: `http://statweb.stanford.edu/~susan/courses/s60/split/node60.html` (visited on 04/28/2020).

[12] Norwegian Meteorological Institute. *Welcome to the MET Norway Weather API.* 2020. URL: `https://api.met.no/` (visited on 05/11/2020).

[13] Alexandros Iosifidis, Anastasios Tefas, and Ioannis Pitas. "DropELM: Fast neural network regularization with Dropout and DropConnect". eng. In: *Neurocomputing* 162 (2015), p. 57. ISSN: 0925-2312.

[14] Adam J Kalkstein et al. "An analysis of air mass effects on rail ridership in three US cities". eng. In: *Journal of Transport Geography* 17.3 (2009), pp. 198–207. ISSN: 0966-6923.

[15] Keras. *Keras API reference.* 2020. URL: `https://keras.io/api/` (visited on 03/03/2020).

[16] Keras. *Keras: The Python Deep Learning library.* 2020. URL: `https://keras.io/` (visited on 04/26/2020).

[17] Pwint Khine and Zhao Wang. "Data lake: a new ideology in big data era". eng. In: vol. 17. Les Ulis: EDP Sciences, 2018. URL: `http://search.proquest.com/docview/2038321379/`.

[18] Norsk Klimaservicesenter. *Hjemmeside.* 2020. URL: `https://klimaservicesenter.no/` (visited on 03/21/2020).

[19] *Learn About Cross Validation in R With Data From the Adult Census Income Dataset (1996).* eng. 2019.

[20] Ding-Fang Li et al. "Fuzzy relevance vector machine for learning from unbalanced data and noise". eng. In: *Pattern Recognition Letters* 29.9 (2008), pp. 1175–1181. ISSN: 0167-8655.

[21] Stephen Marsland. *Machine learning : an algorithmic perspective.* eng. Boca Raton, FL, 2015.

[22] Ali Mesbah and Arie van Deursen. "Migrating Multi-page Web Applications to Single-page AJAX Interfaces". In: (2006).

[23] Divyanshu Mishra. *Regression: An Explanation of Regression Metrics And What Can Go Wrong.* 2019. URL: `https://towardsdatascience.com/regression-an-explanation-of-regression-metrics-and-what-can-go-wrong-a39a9793d914` (visited on 04/29/2020).

[24] Douglas C Montgomery. *Introduction to Linear Regression Analysis.* eng. 5th ed.. Wiley Series in Probability and Statistics Ser. 2012. ISBN: 9781118627365.

[25] Bergen Municipality. *Mobility Lab for Smart Transport Solutions.* 2019. URL: `https://www.bergen.kommune.no/hvaskjer/tema/must` (visited on 02/13/2020).

[26] Bergen Municipality. *Opened lab for future transport.* 2019. URL: `https://www.bergen.kommune.no/hvaskjer/samfunn/opna-lab-for-framtidig-transport` (visited on 02/13/2020).

[27] Bergen Municipality. *Smartere transport i Norge*. 2019. URL: https://www.regjeringen.no/contentassets/827bdf0cac9243dcafd5fa5c12be5456/hordaland_bidrag-til-konkurransen-smartere-transport-i-norge.pdf (visited on 02/25/2020).

[28] Ismail Mustapha and Faisal Saeed. "Bioactive Molecule Prediction Using Extreme Gradient Boosting". eng. In: *Molecules* 21.8 (2016), p. 983. ISSN: 14203049. URL: http://search.proquest.com/docview/1813200659/.

[29] Anuja Nagpal. *L1 and L2 Regularization Methods*. 2017. URL: https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c (visited on 05/03/2020).

[30] Alexey Natekin and Alois Knoll. "Gradient boosting machines, a tutorial". eng. In: *Frontiers in Neurorobotics* 7 (2013). ISSN: 16625218. URL: http://search.proquest.com/docview/2294047772/.

[31] United Nations. *Goal 11: Make cities inclusive, safe, resilient and sustainable*. 2020. URL: https://www.un.org/sustainabledevelopment/cities/ (visited on 05/22/2020).

[32] United Nations. *Goal 13: Take urgent action to combat climate change and its impacts*. 2020. URL: https://www.un.org/sustainabledevelopment/climate-change/ (visited on 05/22/2020).

[33] Shun-Chen Niu. *Data Collection and Sampling. University Lecture*. 2020. URL: https://personal.utdallas.edu/~scniu/OPRE-6301/documents/Data_Collection_and_Sampling.pdf (visited on 05/24/2020).

[34] Ken Peffers et al. "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77. ISSN: 07421222. URL: http://www.jstor.org/stable/40398896.

[35] Anthony Puca et al. *Microsoft Azure*. eng. Apress, 2015. ISBN: 1484210441.

[36] Python. *Installing Python Modules*. 2020. URL: https://docs.python.org/3/installing/index.html (visited on 05/21/2020).

[37] React. *A JavaScript library for building user interfaces*. 2020. URL: https://reactjs.org/ (visited on 05/11/2020).

[38] Bill Schmarzo. *The Data Lake*. eng. Indianapolis, Indiana, USA: John Wiley & Sons, Inc, 2016, pp. 133–152. ISBN: 9781119181118.

[39] Scikit-learn. *About us*. 2020. URL: https://scikit-learn.org/stable/about.html (visited on 05/30/2020).

[40] Scikit-learn. *Sklearn model selection train test split*. 2020. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (visited on 03/10/2020).

[41] Linda Shields and Alison Twycross. "The difference between quantitative and qualitative research". eng. In: *Paediatric Nursing* 15.9 (2003), pp. 24–24. ISSN: 0962-9513.

[42] Abhishek Singhal, Camille Kamga, and Anil Yazici. "Impact of weather on urban transit ridership". eng. In: *Transportation Research Part A: Policy and Practice* 69 (2014), p. 379. ISSN: 0965-8564.

[43] Skyss. *Hordaland won 12,5 millions to smarter transport.* 2018. URL: `https://www.skyss.no/Verdt-a-vite/Nytt-fra-Skyss/vant/` (visited on 02/25/2020).

[44] Skyss. *Om Skyss.* 2020. URL: `https://www.skyss.no/kontakt-oss/om-skyss/` (visited on 04/28/2020).

[45] Skyss. *Vestland er her – kva med Skyss?* 2019. URL: `https://www.skyss.no/Verdt-a-vite/Nytt-fra-Skyss/no-kjem-vestland--kva-med-skyss/` (visited on 04/28/2020).

[46] SSB. *Innbyggere i Bergen.* 2020. URL: `https://www.ssb.no/kommunefakta/bergen` (visited on 03/19/2020).

[47] Manohar Swamynathan. *Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics Using Python.* eng. 2nd ed.. 2019. ISBN: 9781484249475.

[48] Sergios Theodoridis. *Machine learning : a Bayesian and optimization perspective.* eng. London ; 2020.

[49] Contributor to XGBoost Tianqi Chen. *What is the difference between the R gbm (gradient boosting machine) and xgboost (extreme gradient boosting)?* 2015. URL: `https://www.quora.com/What-is-the-difference-between-the-R-gbm-gradient-boosting-machine-and-xgboost-extreme-gradient-boosting/answer/Tianqi-Chen-1?srid=8Ze` (visited on 05/03/2020).

[50] Claes Wohlin et al. "Planning". eng. In: *Experimentation in Software Engineering.* 2012th ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 89–116. ISBN: 9783642290435.

[51] D.H Wolpert and W.G Macready. "No free lunch theorems for optimization". eng. In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 67–82. ISSN: 1089-778X.

[52] Jeffrey M. Wooldridge. *Introductory econometrics : a modern approach.* eng. S.l., 2013.

[53] XGBoost. *Python API Reference.* 2020. URL: `https://xgboost.readthedocs.io/en/latest/python/python_api.html` (visited on 05/20/2020).

[54] XGBoost. *XGBoost Documentation.* 2020. URL: `https://xgboost.readthedocs.io/en/latest/` (visited on 04/30/2020).

[55] Baolin Xu and Chunjing Lin. "An extended practical three-tier architecture based on middleware". eng. In: *2013 IEEE 4th International Conference on Software Engineering and Service Science*. IEEE, 2013, pp. 243–246. ISBN: 9781467349970.

[56] Yr. *Her regner det 200 dager i året*. 2019. URL: `https://www.yr.no/artikkel/her-regner-det-200-dager-i-aret-1.14485816` (visited on 03/17/2020).

Appendix A

Extra Graphs And Plots

A.1 Questionnaire

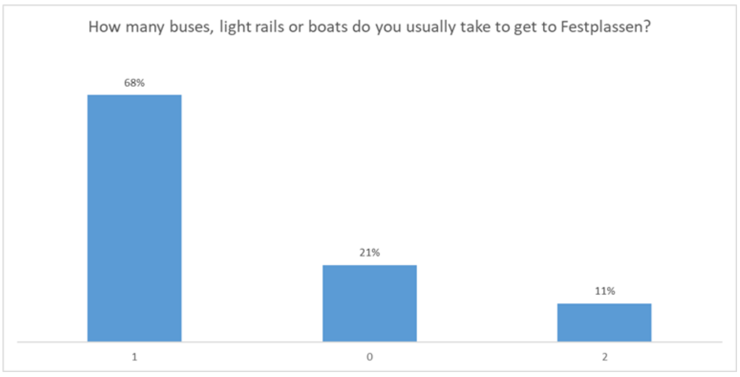


Figure A.1: Number of buses to Festplassen

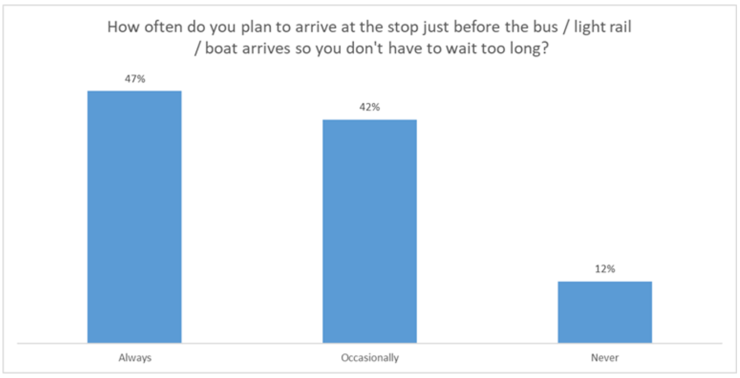


Figure A.2: Plan to arrive at stop

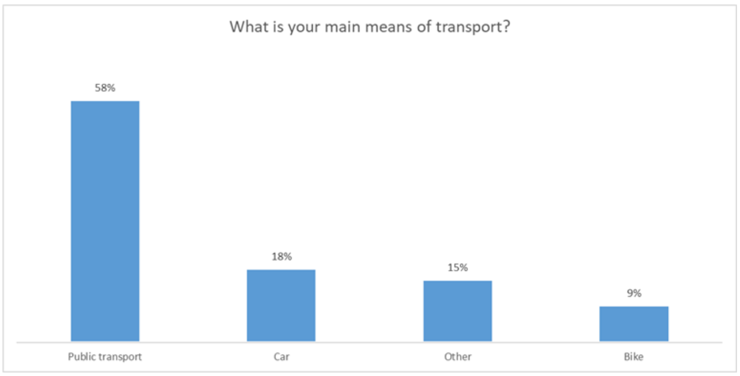


Figure A.3: Main means of transport

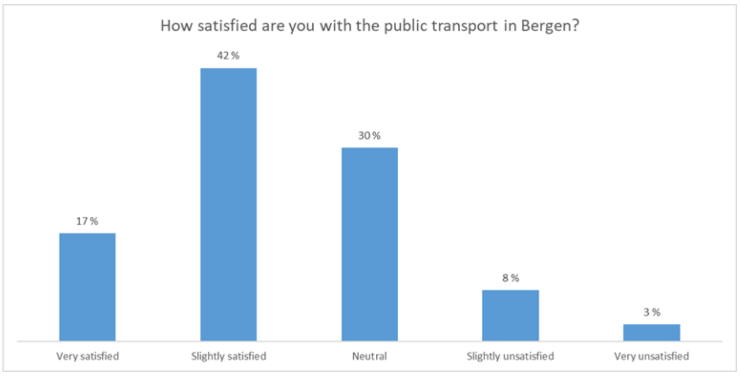


Figure A.4: Satisfied with public transport

A.2 Machine Learning Plots

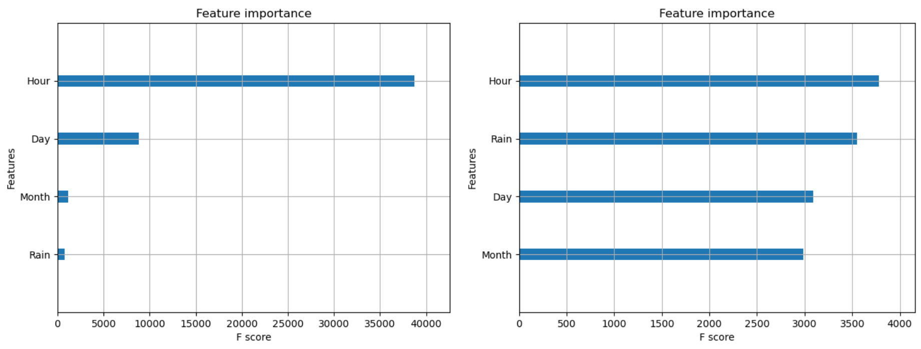


Figure A.5: The feature importance gain and weight for Nonneseter

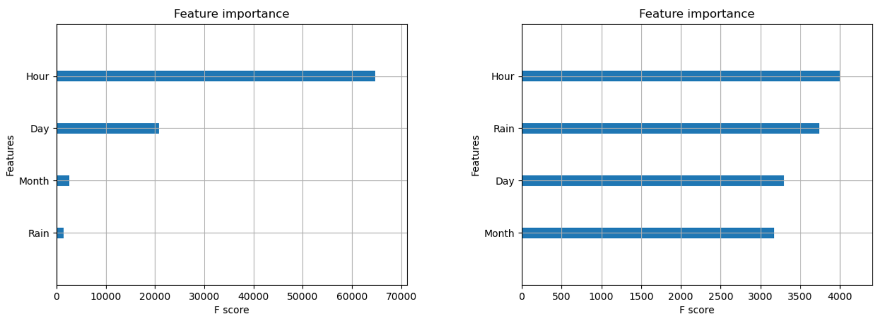


Figure A.6: The feature importance gain and weight for Kronstad