NTNU

Norwegian University of
Science and Technology

Evaluating Machine Learning Methods for City Bike Demand Prediction in Oslo

Lasse Drevland
Patrick Finseth

Master of Science in Informatics
Submission date: June 2018
Supervisor: Trond Aalberg, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

A bike-sharing system is a service in which a fleet of bicycles is made available to the public on a short-term basis through self-served docking stations. These stations are limited in capacity and are often depleted or saturated with bikes due to sudden spikes in demand. These spikes are hard to avoid and are both detrimental to the user experience and the effectiveness of the system. Machine learning methods have been used to forecast demand spikes at station level in similar systems successfully and would likely be a valuable tool in proactively counteracting the effect of demand spikes in the Oslo bike-sharing system.

The goal of this thesis is to evaluate common machine learning methods for demand prediction modeling at individual bike-sharing stations in Oslo.

To accomplish this, four machine learning methods which have successfully predicted station-level demand in similar systems are evaluated through a set of experiments. The methods are based on a *random forest*, *gradient boosting tree* and *recurrent neural networks* with either *long short-term memory*- or *gated recurrent unit* units.

Based on the experimental results, the recurrent neural network with the long short-term memory unit is deemed to be the most suitable for the Oslo bike-sharing system, both due to performance and future potential. However, all methods achieved good performance and made accurate predictions.

The results pave the road for developing a full-scale prediction system in the Oslo bike-sharing system, by highlighting the most promising prediction method.

Sammendrag

Et bysyskkelsystem er en tjeneste hvor en flåte sykkeler er tilgjengeliggjort for offentligheten på kort-tids basis, gjennom selvbetjente låse-stasjoner. Disse stasjonene er begrenset i kapasitet og ofte tomme eller fulle på grunn av sporadisk stor pågang. Dette er vanskelig å unngå og påvirker brukeropplevelsen og effektiviteten i systemet negativt. Maskinlæringsmodeller har vist seg å være gode til å forutse etterspørsel på stasjonsnivå i lignende bysykkelsystem, og kan derfor være et viktig hjelpemiddel i proaktivt arbeid med å unngå effekten av plutselig stor etterspørsel i bysykkelsystemet i Oslo.

Målet med masteroppgaven er å finne ulike maskinlæringsmodeller som ofte er brukt for å forutse etterspørsel i bysykkelsystemer og evaluere disse på bakgrunn av hvor godt de forutser etterspørsel på stasjonsnivå i bysykkelsystemet i Oslo.

Fire maskinlæringsmetoder basert på *recurrent neural networks* med enten *long short-term memory*- eller *gated recurrent unit* minneblokker, *random forest* og *gradient boosting tree*, er implementert og evaluert gjennom tre eksperimenter.

Basert på resultatene er *recurrent neural networks* med *long short-term memory* den mest passende metoden for Oslo bysykkelsystem, da denne har god ytelse og potensiale for videre utvikling. Øvrige metoder viser også god ytelse.

Ved å utheve den mest lovende modellen for å forutse etterspørselen, legges et godt grunnlag for videreutvikling metodene i et fullskala system i Oslo bysykkel.

Preface

The thesis was written autumn 2017 to spring 2018 at the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU).

We would like to thank our supervisor, Trond Aalberg, and Hans Martin Espegren at UIP for their advice and guidance throughout this project. Furthermore, we would like to thank our partners, family, and friends for their support.

Patrick Halmøy Finseth & Lasse Drevland
June 1, 2018
Trondheim, Norway

Table of Contents

List of Tables

List of Figures

Abbreviations

BSS	=	Bike-sharing System
UIP	=	Urban Infrastructure Partner AS
RNN	=	Recurrent Neural Network
RF	=	Random Forest
FNN	=	Feed-forward Neural Network
GBT	=	Gradient Boosting Tree
MAE	=	Mean Absolute Error
MSE	=	Mean Squared Error
L2	=	Least Squares (2)
CART	=	Classification and regression tree
ANN	=	Artificial Neural Network
LSTM	=	Long-short Term Memory
GRU	=	Gated Recurrent Unit
RMSE	=	Root Mean Squared Error
RMSLE	=	Root Mean Squared Logarithmic Error
GCP	=	Google Cloud Platform
SQL	=	Structured Query Language

Chapter 1

Introduction

1.1 Motivation

The world is currently seeing an influx of shared transport services, aided by advances in technology such as the Internet and the modern smartphone.

The concept of the bike-sharing system (BSS) was first introduced more than 50 years ago in the Netherlands (Shaheen et al., 2010) and has recently seen a rapid rise in popularity. Consequently, the amount of systems worldwide has risen from less than a handful in 1998 to more than 800 in 2015 (Fishman, 2016).

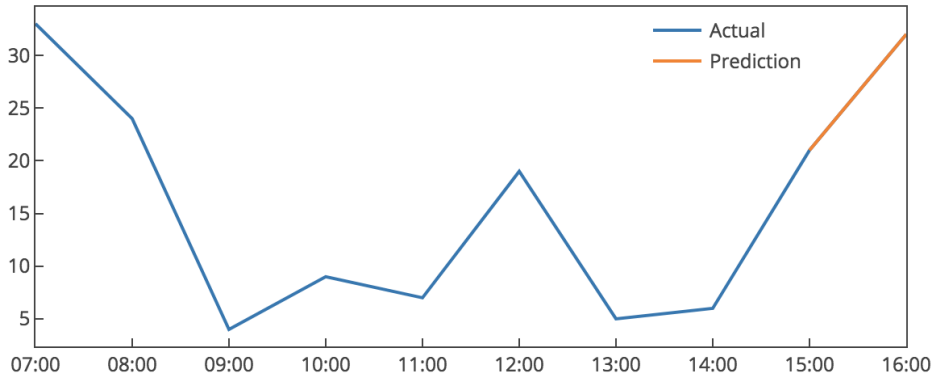
These services are becoming better and more accessible. Users expect to be able to pick up a shared bike at any station at any time and return it whenever they want. However, this is not always the case, sudden surges in demand may deplete or saturate stations completely, leaving users unable to travel as planned. Current solutions to this problem rely on reactive measures and intuition, which can be inefficient and even counter-effective.

An essential element to solving this problem is detailed knowledge about usage patterns. Recent research has shown that it is possible to achieve accurate predictions of demand using a range of machine learning methods In this thesis, the aim is to evaluate common machine learning methods for demand prediction at individual bike-sharing stations in Oslo through a set of experiments. The demand-patterns are highly dependent on a lot of factors such as time of day, weather, season and the inventory of neighboring stations, and sudden influxes in demand can happen seemingly at random.

Forecasting these patterns is, therefore, a burgeoning topic (Fishman, 2016), as an accurate prediction could have a significant impact on the quality of the system. Knowing the size and timing of these influxes could be utilized to warn users that the station they are biking to is filling up, and it would allow proactive effort in counteracting depletion and saturation. It could also provide the system operators with valuable insight that would allow them to modify the system to avoid these influxes entirely.

1.2 Problem formulation

The task of predicting future demand can be classified as a time series prediction problem. Time series prediction is the task of predicting the future given a series of historical observations. The goal of this thesis is to use state-of-the-art research about time series prediction and machine learning to evaluate machine learning methods for station-level demand prediction in the Oslo BSS, with demand defined as the number of started trips at a given station within a specific period.



Demand for a station spanning nine hours with a one hour sample interval and prediction-span.

Figure 1.1: Example of station-level BSS demand

Figure 1.1 shows an example of demand for a single station. Historic data is colored blue, and is used to train the machine learning methods. The goal is to predict the next period which in the figure is colored orange.

The Oslo BSS has a different amount of data and a topology that may present other dynamics than those of related BSSs, which might favor other methods than those who have performed well in related research.

Research Goal

The goal of this thesis is to evaluate the performance of common machine learning methods for predicting station-level BSS demand in the Oslo BSS.

Research Question 1

Which commonly used machine learning methods are the best candidates for predicting station-level demand in the Oslo BSS?

A range of machine-learning methods has been successfully used to predict station-level demand in other bike-sharing systems. These differ from the Oslo BSS in important aspects that influence demand and might alter the performance outcome. Therefore, it is essential to find methods that have achieved good results on systems that closely resemble the Oslo BSS.

Research Question 2

How accurate are these methods in predicting demand in the Oslo BSS?

The accuracy of the machine-learning methods determines how valuable they are to the stakeholders in the BSS.

Research Question 3

How does the performance of these methods differ?

Differentiating the methods is important in order to determine what methods to utilize given a criterion.

1.3 Approach

A three-step process is proposed to fulfill the research goal. First, the state-of-the-art in BSS demand prediction and -analysis is examined to find the most promising methods for the Oslo BSS data. Second, the most promising methods are run through a set of experiments with the Oslo BSS data, designed to accentuate key performance aspects and provide a reliable basis for evaluation. Third, the experiment results are evaluated with emphasis on differences that are of importance to the stakeholders in the Oslo BSS.

1.4 Contributions

This research contributes to the field of BSS by establishing the most suitable machine learning model for station-level demand prediction in the Oslo BSS. It further shows that all selected methods can be used to achieve good predictions on a 30- and 60-minute prediction-span. Additionally, it shows that extrapolation as a method to account for missing data and censored demand distorts the demand patterns too much, and more careful calculations are needed to successfully use this to predict estimated demand.

1.5 Overview

Chapter 1: Introduction

The introduction provides a brief overview of the contents of this thesis.

Chapter 2: Background

The background presents knowledge necessary to understand the context and theory of the problem, in addition to an examination of related work and an analysis of the available data.

Chapter 3: Theory

The theory gives an in-depth description of the machine learning methods utilized, as well as an overview of relevant performance metrics and validation methods.

Chapter 4: Method

The method describes the research method, the tools used and the process of designing and preparing the experiments.

Chapter 5: Experiments

The experiments provides a step-wise walk-through of the experiments.

Chapter 6: Results and discussion

The results and discussions chapter outlines the results, presents an in-depth evaluation of the experiment results and describes how this resolve the research questions.

Chapter 7: Conclusion

The conclusion concludes this research and its limitations and presents thoughts on essential directions for future work.

Chapter 2

Background

2.1 Bike-sharing systems

A bike-sharing system is a service in which a fleet of shared bicycles is made available to the public on a short-term basis through self-served docking stations. The bikes may be rented and returned to any of the docking stations included in the system. The rental duration is commonly kept in the 30-45 minute range, to maximize the number of rides per bike. Often the goal of introducing these systems is to provide a sustainable and flexible alternative to commuting in urban areas to reduce congestion and pollution caused by private cars.

The shared bicycle programs are commonly divided into four generations, the first of which, *free bike systems*, were free to use and consisted of a set of brightly colored bicycles scattered throughout the cities in which they were deployed. They were plagued with theft, and vandalism and most programs did not last long. The next generation, known as *coin-deposit systems* saw its inception in Copenhagen in 1995 and introduced docking-stations where a small deposit was required to release the bikes. There were no time limits on rental and the deposit amount was small, so the bikes were often used for excessively long periods or even not returned at all. The incorporation of docking-stations made the system more dependable and resistant to theft but increased the cost of operation.

In 1998 the first third-generation system appeared in France with the distinguishing feature of incorporating IT-systems for administration and tracking. This facilitated the usage of smart-cards or credit-cards to rent bikes, and made it possible to study real-time usage data of the system on a large scale.

Most bike-sharing systems today operate at the third generation, and the fourth as proposed by Shaheen et al. (2010) is rolling out across the world.

Bike sharing systems have gained popularity as a new domain for machine learning due to the massive amounts of data collected by the systems (Jia et al., 2017), in addition to its ability to capture temporal movement patterns in cities. Much can be learned from this, and bike-sharing programs often encourage research by offering the data free of charge.

2.2 Oslo bike-sharing system

The first BSS in Oslo was established in 2002 by Clear Channel and continued in operation until the end of 2015. Upon which Urban Infrastructure Partner (UIP) took over management and revamped the service for launch in early 2016.

As of May 2018, the Oslo BSS consists of about 3000 bicycles and 205 stations. Bikes can be rented from any station for up to 45 minutes at a time, after which a small fee is charged for exceeding rental limits. The fee is charged unless one is close to a full station and unable to return the bike. In this case, one is granted 15 additional minutes for free to find an available spot at another station.

To rent a bike, one need to have purchased any of two rental plans, for either a season or a day with a price of 399 NOK and 49 NOK respectively. The users primarily interact with the system through the Oslo bike sharing application in which one can check the availability of locks and bikes at nearby stations, report broken bicycles, enable extended rental and unlock bikes. The latter can also be accomplished through a touchscreen interface at each station.

The system is closed throughout the winter as the weather conditions become unsuitable for biking. In 2016 and 2017 the system opened in early April and closed around early December. Additionally, the system is also closed at night between midnight and six in the morning. The current generation of bikes and stations are specially built for shared bicycle systems, with vandal-proofing and optimization for low maintenance. The system is growing continuously and has increased considerably in extent between 2016 and 2018.

2.3 Problem variations

2.3.1 Demand granularity

BSS demand forecasting is commonly divided into three types, where each presents a different difficulty and value. With the demands defined as the number of started or ended trips within a specific period, the problem is split into the following categories.

System-level demand
: The total demand of all stations combined.

Clustered demand
: The demand in a cluster of stations. Where the clustering is based on factors such as similarities in demand profile or physical distance.

Station-level demand
: The demand at an individual station.

2.3.2 Demand types

BSS demand can additionally be split by the following aspects of demand, which, however, typically share characteristics in the system.

Started trips
: The number of started trips at a station within a specific period.

Ended trips
: The number of ended trips at a station within a specific period.

Net trips
: The difference in the number of started and ended trips at a station within a specific period.

2.4 Related work in bike-sharing system demand

The topic of bike-sharing is burgeoning with research on many aspects in addition to demand prediction, such as usage and user preferences, history and growth, re-balancing, barriers to bike-sharing and future directions (Fishman, 2016). In the following sections, five successful works in demand modeling and prediction are presented and compared in depth. Other work that to some degree has inspired and directed this work is listed with a short annotation of their contents.

The papers that are examined represent state-of-the-art within their aspects of bike sharing research and are selected based on likeness to the Oslo BSS regarding the data available. Rudloff and Lackner (2014) aimed to improve redistribution efficiency by attaining better station-level demand predictions using more features than previous efforts. The research was combined with a feature study using a set of count models. Chen et al. (2017) believed that an efficient *recurrent neural network* (RNN) based method could outperform traditional methods such as *random forest* (RF) and *gradient boosted tree* (GBT) on station-level and system-level demand prediction. They make predictions for started and ended trips at all stations as well as system and station-level demand in a combined model.

Froehlich et al. (2009) examined a multitude of different aspects of the BSS, including inferring cultural and geographical features of the host city, demand forecasting at station-level and predictions of station-level inventory. Tran et al. (2015) modeled bike-sharing demand at station-level using linear regression to examine the effect of environmental variables on demand. Yin et al. (2012) compared several machine learning techniques on a simplified problem to predict system-level demand.

2.4.1 Datasets

The datasets used in these studies originate from different BSSs around the world. Rudloff and Lackner (2014) used a dataset sourced from the BSS Vienna City Bike in Austria, containing trip records collected over a 2-year period spanning from 2010 to 2012. The dataset was supplemented with station inventory status sampled every 15 minutes and weather data for every station.

Chen et al. (2017) performed their experiments using trip data recorded in 2014 in the New York Citi Bike BSS, with more than 8 million trips. Froehlich et al. (2009) attained their data by sampling it from the website of Bicing Barcelona every two minutes to accumulate 26 million individual data points, however, this data only covered a span of 13 weeks in 2008 from the August to December and was prone to errors due to the collection method.

Tran et al. (2015) collected 6 million trips conducted in 2011 from the BSS of Lyon in France. Yin et al. (2012) used the smallest dataset of the five, containing 17 thousand data points of system status for the Capital Bikeshare BSS in 2011 and 2012. Both Rudloff and Lackner (2014) and Chen et al. (2017) noted a seasonal change in demand where usage peaks in the late summer and is reduced during the winter.

2.4.2 Method

Numerous methods of predicting demand can be employed and these works display a specter of the most successful. All papers concerned with predictions, model individual stations, aside from Yin et al. (2012) who modeled the total demand at system-level.

Rudloff and Lackner (2014) found that predictions of stations modeled individually yields good predictions, and Chen et al. (2017) noted that this would be more useful to stakeholders. Chen et al. (2017) bench-marked their proposed *recurrent neural network* (RNN) with a few variations against *ordinary least-square regression*, *random forest* and a *feed-forward neural network* (FNN).

Yin et al. (2012) developed models using *ridge regression*, *support vector regression*, *random forest* and *gradient boosting tree* (GBT). Wherein the last two achieved the best predictions. Froehlich et al. (2009) employed a much simpler approach, using simple predictive methods such as *last value*, *historic mean*, *historical trends* and a *Bayesian network* to create the forecasts.

2.4.3 Features

In addition to the variety of methods that can be applied, there can be a considerable difference among BSSs in what features determine demand. The variety can depend on anything from work schedules to topography, however, most found the weather to be among the most important.

Specifically, Rudloff and Lackner (2014) and Chen et al. (2017) both found that there exists a strong positive correlation between ridership and temperature and a weaker negative correlation with precipitation. Yin et al. (2012) ranked weather as the most crucial variable after hour, working day, season, and year.

Tran et al. (2015) found that public transport and more specifically train arrivals was a significant instigator of demand. Furthermore, they found that variables such as station altitude, population numbers, proximity to leisure locations and the density of the BSS also affect the demand profile. Lastly, the inventory status of neighboring stations was noted as a significant feature in determining demand by Rudloff and Lackner (2014).

2.4.4 Prediction-span

Despite examining multiple aspects and utilizing different methods, all the studies found that predicting and modeling BSS demand at 60-minute intervals made the most sense and provided the best results. Froehlich et al. (2009) examined the entire span from 10 minutes to 120 minutes as this seemed to correspond best with the usage patterns. Rudloff and Lackner (2014) argued that a 60-minute period would be enough to provide reasonable value while avoiding prolonged periods of zero demand which would be detrimental to the results.

2.4.5 Results

All the studies referenced in this section achieved excellent results in their respective tasks. Rudloff and Lackner (2014) noted that introducing new stations close to existing ones had a noticeable effect on the demand and should be accounted for when forecasting. The RNN model developed by Chen et al. (2017) outclassed the benchmark algorithms both globally and locally.

Froehlich et al. (2009) managed to create good predictive models using only a few weeks worth of data. Additionally, it was found that stations with higher activity proved more difficult to predict and that prediction quality was a function of the size of the prediction-span.

Tran et al. (2015) found a big difference in the demand patterns caused by long-time and short-time subscribers. Finally, RF and GBT achieved the best performance on global system demand in the study conducted by Yin et al. (2012).

2.4.6 Other research

2.4.6.1 Reviews

- **Bike-Sharing System: A Big-Data Perspective. Jia et al. (2017).** Literature comparison and analysis of Big-Data papers concerned with BSS planning, pattern analysis, demand or trip prediction and re-positioning.

- **A Review of Recent Literature. Fishman (2016).**
  Review of BSS literature conducted since 2013 concerned with general aspects such as history, growth, usage patterns, user preferences and demographics.

- **A Synthesis of the Literature. Fishman et al. (2013).**
  An evaluation of the global state of BSS as of 2013.

2.4.6.2 Prediction

- **Predicting Bike Usage for New York City's Bike Sharing System. Singhvi et al. (2015).** Pairwise trip-demand prediction using regression methods at neighbourhood-level in New York City's Citi Bike system. Covariates include taxi usage and weather.

- **Bicycle sharing systems demand. Frade and Ribeiro (2013).** Relating BSS demand to external factors that affect usage, in order to estimate demand in locations without an existing BSS.

- **Bicycle-Sharing System Analysis and Trip Prediction. Zhang et al. (2016).** Bike-Sharing trip prediction with geographical data in addition to customer features in order to provide input for better re-balancing.

2.4.6.3 Other BSS challenges

- **Understanding Bike-Sharing Systems using Data Mining: Exploring Activity Patterns. Vogel and Mattfeld (2011).** Data mining used to gain insight into BSS activity patterns in order to attain a better understanding of the system structure.

- **Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. Kaltenbrunner et al. (2010).** A spatio-temporal pattern analysis of the Bicing BSS in Barcelona using data sampled from the system website, to provide predict system-level availability in a short time span.

- **Station Site Optimization in Bike Sharing Systems. Liu et al. (2015).** An artificial neural network model is built to predict station demand and balance, and is used as input to a station site optimization algorithm that aims to improve system balance.

2.4.6.4 Other domains

- **DeepSD: Supply-Demand Prediction for Online Car-hailing Services using Deep Neural Networks. Wang et al. (2017).** Development of a *deep residual neural network* that outperform previous efforts in demand-supply prediction with car-hailing services.

- **A new Evolutionary Neural Network for forecasting net flow of a car sharing system. Xu and Lim (2007).** An *evolutionary neural network* optimized with a mixed genetic algorithm and back-propagation was utilized to predict the net flow of cars in a car sharing system in Singapore.

2.4.7 Summary of related work

These works contain numerous exciting aspects that influence this work. Though the quality and quantity of the datasets were varying, one year of data seemed to be enough to achieve good predictions, and even a few weeks can be sufficient. The most commonly used machine learning methods, in no particular order, are the following:

- Feed Forward Neural Network (FNN)
- Recurrent Neural Network (RNN)
- Evolutionary Neural Network
- Residual Neural Network
- Random Forest (RF)
- Gradient Boosting Tree (GBT)
- Super Vector Regression
- Ridge Regression

The weather was decidedly the most critical variable in all studies where it was included, wherein temperature and precipitation correlated the most. Station- and neighbor station inventory also proved valuable. A 60-minute prediction-span was found to provide the most value without compromising prediction quality. These studies show that though there are many factors influencing prediction quality such as activity level and changes to system architecture, good results can be achieved in predicting BSS demand using relatively sparse data and few features. The most promising methods for the Oslo BSS regarding demand prediction is the RNN, RF, and GBT, as these achieved excellent results using data that is similar to what is available for this problem.

2.5 Datasets

The following datasets are utilized in this research. The trip records and station details are provided by UIP, and the historical weather data is downloaded from the Norwegian Meteorological Institutes data-portal (eKlima) [1].

- Trip records (UIP)
  - Censored demand
  - Estimated demand
- Station details (UIP)
- Historical weather (The Norwegian Meteorological Institute)

2.5.1 Trip records dataset

The trip records dataset contains records of trips conducted during 2016 and 2017, with detailed information about every trip conducted within the BSS. Each record lists which station the bike was rented from and to which it was returned, the duration of the start and a timestamp for when the trip started and ended. It further includes subscription id, bike id, and other contextual data.

Bikes that have been moved manually by the system operators are not recorded in this dataset, but it does, however, include some test data such as trips conducted during the night and trips including internal stations. This data is filtered during preprocessing.

In this thesis, the demand is split into the two measures *estimated-* and *censored demand*. This is done to account for situations where the stations are depleted, and no trips can be initiated from the station.

[1] www.eklima.met.no

The demand is considered to be censored when a station is depleted and no trips are recorded, despite there possibly being a demand for bikes. Estimated demand is an extrapolation of the censored demand for bikes for the time in which it is censored.

Table 2.1 shows the most relevant columns of the trip records dataset, and table 2.2 shows the columns in the estimated demand dataset.

Attribute	Data type
Bike ID	Integer
Started timestamp	Timestamp
Ended timestamp	Timestamp
Station ID for start station	Integer
Station ID for end station	Integer
Status	String
Ended method	String

Table 2.1: Censored demand dataset overview

Attribute	Data type
Timestamp	Timestamp
Demand bikes	Float
Demand locks	Float
Net demand	Float

Table 2.2: Estimated demand dataset overview

2.5.2 Station details dataset

The station details dataset contains information about every station in the Oslo BSS. This includes the id and name of the station, textual placement description, coordinates, elevation above sea-level and the number of bike slots. Figure 2.3 shows the format of each column.

Attribute	Data type
Station ID	Integer
Station title	String
Latitude	Float
Longitude	Float
Elevation	Integer
Slot count	Integer

Table 2.3: Station details dataset overview

2.5.3 Weather dataset

The weather dataset contains hourly weather data recorded at Blindern in Oslo, composed of the four attributes, precipitation, duration of sunshine, wind speed and temperature. Table 2.4 shows the format of each attribute. The weather data should ideally be recorded as close to each station as possible. However, to avoid the increased complexity of collecting and matching weather data for every station, this dataset is used for all stations. The weather data is still be relevant for all stations as it represents the overall conditions in Oslo.

Attribute	Data type
Timestamp	Timestamp
Millimeter of rain last hour	Float
Minutes of sun last hour	Float
Average wind speed last hour	Float
Average temperature last hour	Float

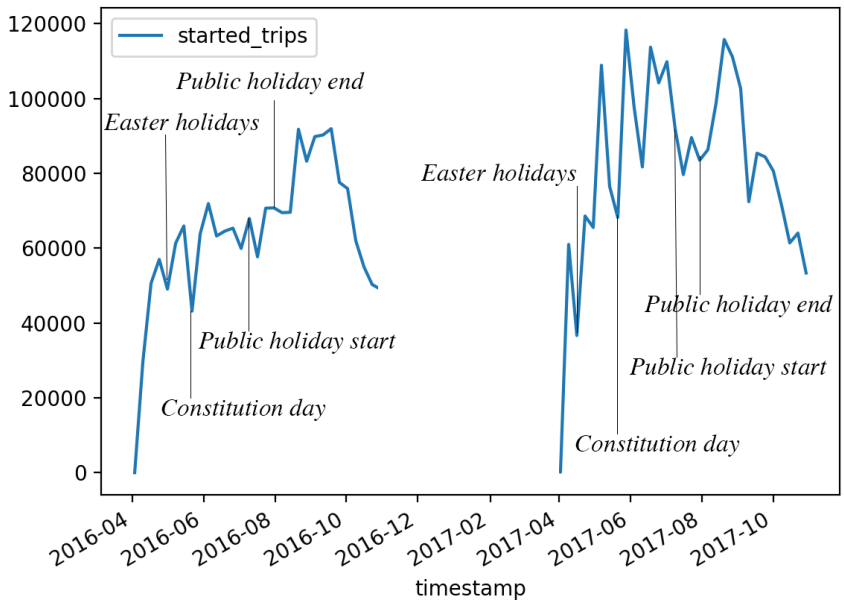
Table 2.4: Historical weather dataset overview

2.6 Data analysis

Data analysis is a crucial step in creating time series forecasting methods. The goal is to identify features in the dataset and determine which can affect the predicted value as well as the significance of these.

The Oslo BSS shares many features with the datasets used in the papers examined in section 2.4 such as seasonality, size, detail, and correlations.

The Oslo BSS under the management of UIP opened at the beginning of April 2016 and has operated for two seasons with a halt in the winter of 2016-17. The BSS opens for the season around the beginning of April and closes in November and has amassed around 4.8 million trips since the launch.



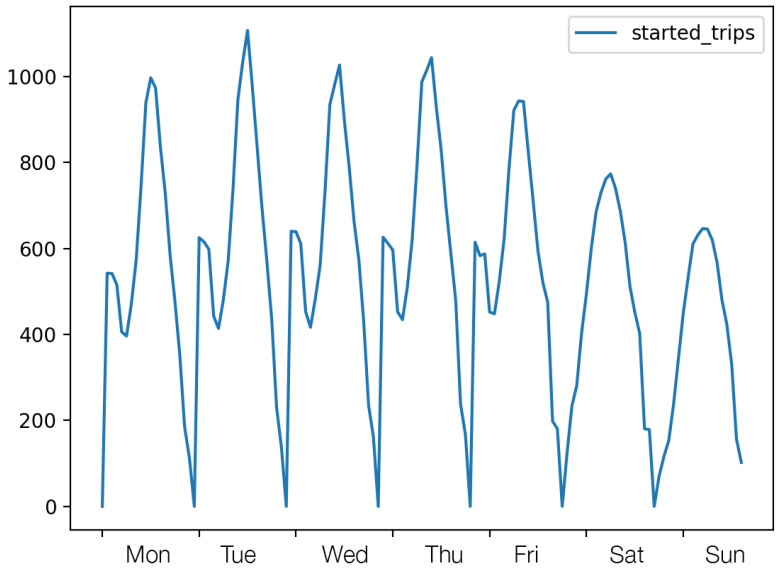
The annotated major public holidays show that the ride volume drops significantly during these periods, and is an indication that ride-volume is closely linked with work schedules.

Figure 2.1: System-level ride volume with public holidays annotated

Figure 2.1 show that usage in both seasons, increases towards the end of the year, with a peak in August and September, and a sharp decrease in November as the winter sets in and the system closes. This trend is also observed by Chen et al. (2017) and Froehlich et al. (2009) in their research. A decrease in volume is observed during public holidays, which can indicate that a large part of the user base is commuters that use shared bikes to get to work.

The 2017 season saw an increased ride-volume up half a million from 2.1 million trips in 2016 to 2.6 million which in part can be attributed to the increased system capacity due to a continuous extension with more bikes and stations. This is shown in figure 2.1.

Figure 2.2 show that weekdays follow a familiar pattern with a sharp peak in the morning between 6:00 and 7:00 and another sharp peak in the afternoon between 15:00 and 16:00. There is no morning peak on weekends, and the afternoon peak is smaller, happens one hour sooner and is not as sharp as the weekday peaks. The Sunday peak is also smaller than the Saturday peak.

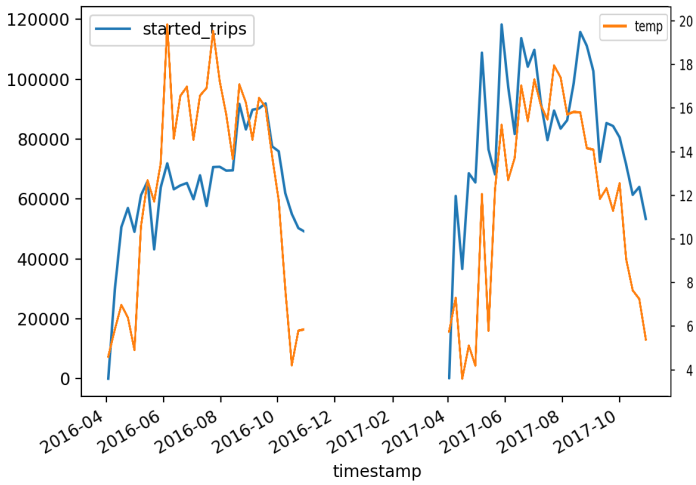


Censored demand aggregated by hour and day of week. Weekdays follow a common pattern with a sharp morning spike followed by a larger and longer spike in the afternoon. There is a generally lower demand and no morning peak on weekends.

Figure 2.2: System-level aggregated censored demand

This pattern further strengthens the hypothesis that commuters make up a big part of the users, as the weekday peaks correspond with normal Norwegian work hours and the fact that the morning peak is absent on weekends. Froehlich et al. (2009), Rudloff and Lackner (2014) and Chen et al. (2017) found similar patterns in the BSSs they studied.

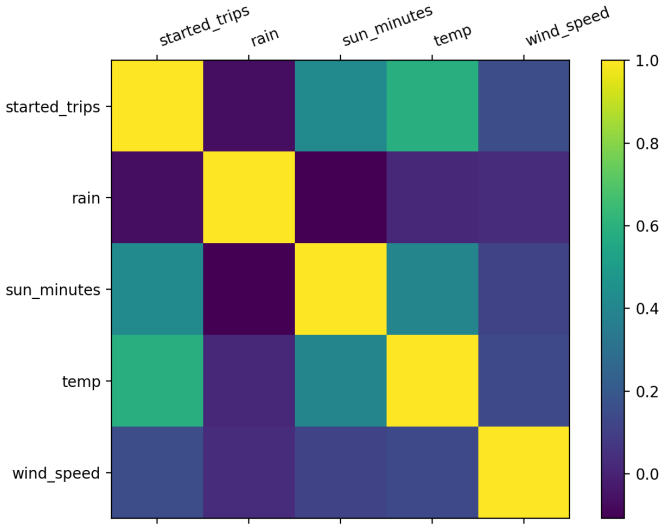
Figure 2.3 show that the traffic volume follows the seasonal change in temperature indicating that it may have a strong correlation with ridership.



System-level ride volume corresponds with the seasonal changes in temperature which indicates that temperature is an important determinant for demand in the Oslo BSS.

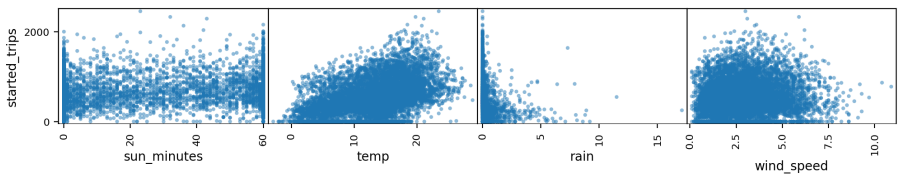
Figure 2.3: Comparison of censored demand and average temperature

Figure 2.4 shows that there is a positive correlation between the temperature and the number of initiated trips, and figure 2.5 shows that there exists a weak linear relationship. The number of minutes with sun also correlates positively with the number of rides. Wind speed has a deficient correlation factor, and figure 2.5 show that a moderate breeze is only slightly deferred ridership. Rain displays a weak negative non-linear correlation that is weaker than what is found in related work (Yin et al., 2012; Rudloff and Lackner, 2014).



The variables ranked by correlation with the amount of started trips, from highest positive- to highest negative correlation is as follows; temperature, minutes of sun, windspeed and then rain.

Figure 2.4: Correlation matrix of average weather and aggregated censored demand



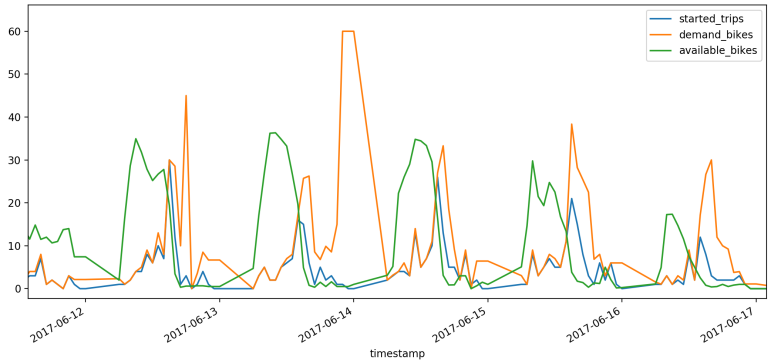
The amount of sun shows zero correlation. Temperature displays a weak positive linear correlation. Rain shows a non-linear negative correlation, and the wind speed shows little or no visible correlation with rideship.

Figure 2.5: Scatter plot of average weather and aggregated censored demand

2.6.1 Estimated demand

The profile of the estimated demand is much more erratic and noisy than the censored demand. The way it is calculated makes it sensitive to situations where a set of bikes are rented within a short interval, causing sudden depletion, leading to false spikes.

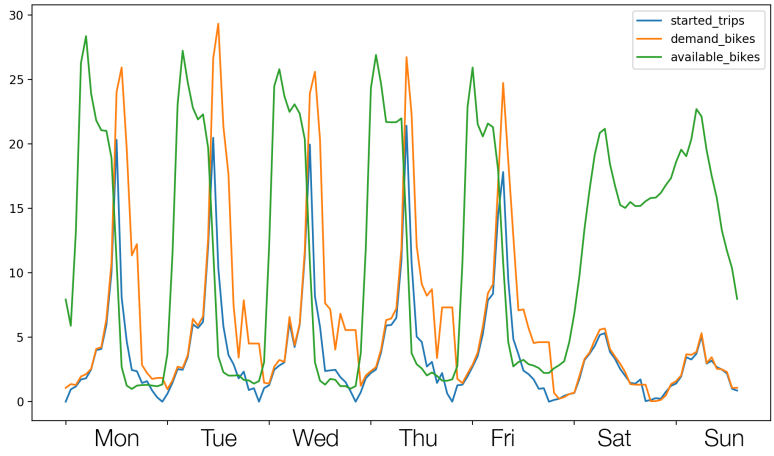
An example of this can be seen in figure 2.6. On Wednesday, June 14, the estimation indicates that there would have been a secondary spike in the afternoon, twice as big as the one before it. The secondary spike is a highly unlikely scenario and indeed an anomaly considering the regular patterns. This problem has been somewhat remedied by limiting the upper bounds of the demand, but cannot be removed entirely as it still might be a result of real demand.



Station inventory level, censored- and estimated demand for station 279 between June 12. and June 17. 2017. The estimated demand is denoted *demand_bikes* and is observed to constantly rises above the censored demand, denoted *started_trips*, just as the station is depleted of bikes. An example of what is probably a false spike is visible on June 14.

Figure 2.6: Comparison of estimated- and censored demand

Figure 2.7 show the demand aggregated by the hour and day of week. Here the demand consistently exceeds the number of started trips on the weekdays, but follow on weekends. This concurs with the fact that the weekend spikes happen over a more extended period and does not deplete the station as rapidly, which indicates that the weekend demand is rarely censored.



Comparing the hourly aggregated censored demand to the estimated demand for station 279 show that what is observed in figure 2.6 is present through this dataset. The estimated demand constantly exceeds the number of started trips on weekdays, indicating that the demand is censored and that the real demand commonly is higher. On weekends estimated demand follows the censored demand, and the station inventory is generally high throughout the day. This indicates that demand is rarely censored.

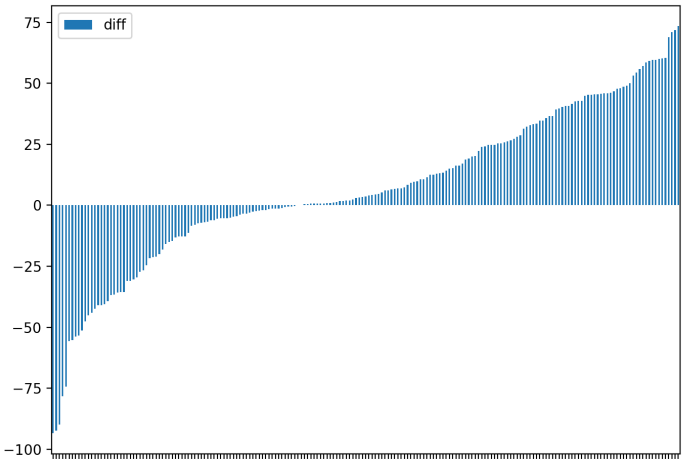
Figure 2.7: Comparison of aggregated estimated- and censored demand

The small secondary afternoon surge observed on Tuesday, Wednesday and Thursday indicate that the demand in this period often is censored and that the number of started trips in this period would have been more substantial had the station not been depleted. It might also be a result of the anomalies of the type described earlier.

2.6.2 Station variations

There are many differences between the 198 stations in the Oslo BSS. They vary in bike lock capacity from six to sixty. The amount of data measured in the number of trips per station, ranges from just a couple at *Trelastgata* to over a hundred thousand at *Alexander Kiellands Plass*. They also differ in the duration of operation, *Trelastgata* opened in November 2017 and has rarely been used, while *Alexander Kiellands Plass* has been operational since the system launch in 2016.

Additionally, there is also a difference in usage patterns. Figure 2.8 show the distribution of the difference in the number of started and ended trips per station in total. A pronounced imbalance in the number of arrivals and departures can be observed for most stations. An example of this are the stations *Bankplassen* and *St. Hanshaugen park vest*, the first can be classified as an arrival-station as it has about half as many departures as arrivals, whereas the latter can be classified as a departure-station as it has twice as many departures as arrivals.



The difference in percentages shows that the difference between the amount of started- and ended trips on a station spans a large range.

Figure 2.8: Distribution of station departure and arrival difference

2.7 Research value for partners

The outcome of this research provides UIP, with information that opens up a range of interesting possibilities for solutions that benefit all stakeholders in the Oslo BSS and other systems UIP operate.

The most beneficial in the short term is probably the possibility to preemptively counteract depletion of stations, by extending the most promising of the evaluated methods. As these situations are disruptive and have a substantial negative impact on user experience, better handling would help improve overall user satisfaction. On a longer horizon, this helps scale and improve the system structure to counteract these situations permanently.

The project could be further utilized to build a system that warns users when nearby stations are expecting a significant surge, and thus even out the demand by letting them select less crowded stations in advance. This has the added benefit of allowing users to more easily learn the demand-patterns of the system and better plan their future usage.

The predictive models are developed with UIPs system architecture in mind which allows for natural extension and implementation with many of their existing applications. Building on Google Cloud Platform facilitates extension by allowing the experimental models to be set up for continuous learning and be used in real time. The models are also fairly general and can be adapted to provide predictions for any of the systems UIP operate provided that there exist a sufficient data basis.

Chapter 3

Theory

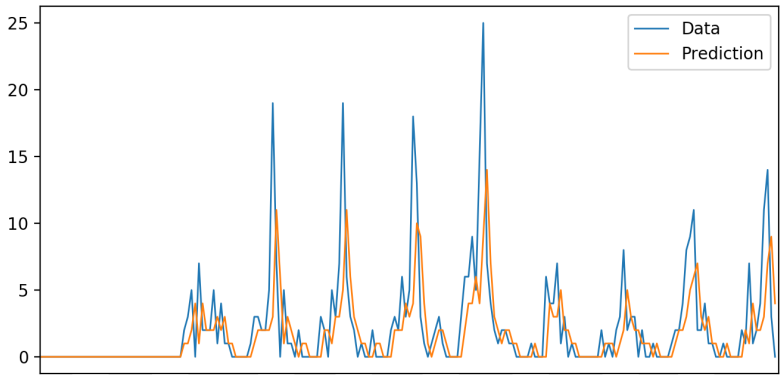
3.1 Time series analysis

The urge to know what happens in the future has driven research on predictions for years. Recently this has accelerated due to increased access to data. Smart watches, refrigerators, cars, and BSSs like this one records vast amounts of data every day, which enable us to predict the future by learning from the past.

Time series analysis is the science of examining a sequence of data points, often collected successively at a set interval to extract information. It can be divided into the three sub-goals; prediction, modeling, and characterization.

The goal of prediction is to forecast the short-term evolution in a system. For modeling, the goal is to find features that describe the long-term behavior, while characterization is the task of determining properties in the system such as the amount of randomness (Weigend, 1994).

A common problem that occurs in time series prediction is naïve forecasting. This occurs when the most robust pattern learned by a prediction model is that the target value follows the last known value. Naïve forecasting is a major pitfall when making predictions using time series data, as it may look like the model has learned the data, while it only relies on the previous value. Figure 3.1 shows a prediction graph with good performance metrics, but a noticeable naive forecast. This shows the importance of evaluation when analyzing time series predictions. A naive forecast is recognized by the prediction graph it produces, which are discovered through a visual inspection of graphs during the optimization step is necessary.



A naïve prediction is characterized by a prediction graph that is offset by the prediction-span from the actual data.

Figure 3.1: Example of naïve prediction

Time series data can be classified according to behavior, and the models are split by linearity (Weigend, 1994).

3.1.1 Linear models

Linear time series models are commonly employed to discover covariance structures, they are easy to implement and can give a good understanding of behavior. The linear models are predominantly divided into *moving average* (MA) and *autoregressive* (AR) models, but combinations of these models such as ARMA and ARIMA also exist (Weigend, 1994). However, time series may exhibit more complex patterns, and linear models are often not enough to capture the more intricate relationships between the covariates.

3.1.2 Nonlinear models

The differentiating feature of nonlinear models is their ability to capture non-linear covariate behavior which allows them to be applied to a broader set of problems. However, this also has the downside of potentially modeling extraneous noise (Weigend, 1994) and makes problem specific tuning and adjustment a more complicated task. This thesis employs the non-linear machine learning models; artificial neural networks, random forests, and gradient boosting trees.

3.2 Machine learning

Machine learning is a branch of artificial intelligence, where a central part of the research is conducted on the ability to automatically learn complex patterns and perform intelligent decisions based on data.

Businesses in a wide range of industries are embracing artificial intelligence and machine learning. It is being used to automate tasks and solve complicated problems previously only a highly educated person could solve. A lot of everyday items are being enhanced with artificial intelligence which now powers autonomous cars, online newspapers, and social networks. We are even seeing the adoption of smart, learning, virtual assistants.

Machine learning employs problem-solving agents that can be simple reflex-, model-based-, goal-based-, utility-based- or learning agents. Learning agents are often used to solve classification, clustering and regression problems and are what is utilized in this thesis. Learning agents have the advantage of being able to operate in unknown environments and become more competent than initial knowledge might allow (Norvig and Russell, 2009).

Reinforcement-, *unsupervised-* and *supervised learning* are learning methods that machine learning models can implement. Using reinforcement learning the agent expands its knowledge based on reward and punishment. In unsupervised learning the agent attempts to find patterns in the data without any feedback from the environment, while in supervised learning the agent learns by comparing its output with the ground truth.

In unsupervised learning, the agent is not provided with target values, which means that it has to determine how to weigh each of the input features based on the given criteria. The primary applications of unsupervised learning are classification tasks where the goal is to describe hidden structures in data without a label. Therefore, there is also no way to evaluate the accuracy of this model (Norvig and Russell, 2009).

In supervised learning, *backpropagation* is used to train the model. The network output is compared to the target value creating an error, which is propagated backward throughout the network to adjust the internal weights. The error is created for every input and target value pair in the training-set until the data function has been learned (Norvig and Russell, 2009). Tasks such as BSS demand prediction are often solved using supervised learning methods, as time series easily can be transformed into input-output pairs due to the ground truth simply being a following value.

Machine learning methods use loss functions to minimize the error between the predicted value and the target value while learning. It produces a real numbered error representing the inconsistency between these values, where a decrease in loss indicates an increase in accuracy. The type of loss function to use depends on many factors such as the type of problem and the method employed. For regression problems three of the most commonly used functions are *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE) and *Least Squares* (L2).

The specific loss function to use is selected through the optimization process, where each function is evaluated to find the most suitable for a given problem. MAE measures the average accuracy for two continuous variables without considering the direction by using the absolute value. MSE and L2 measure the average accuracy by quadratic errors. MAE is more stable for outliers than MSE and L2. Outliers are data points far from the center data mass in a dataset.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.1}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.2}$$

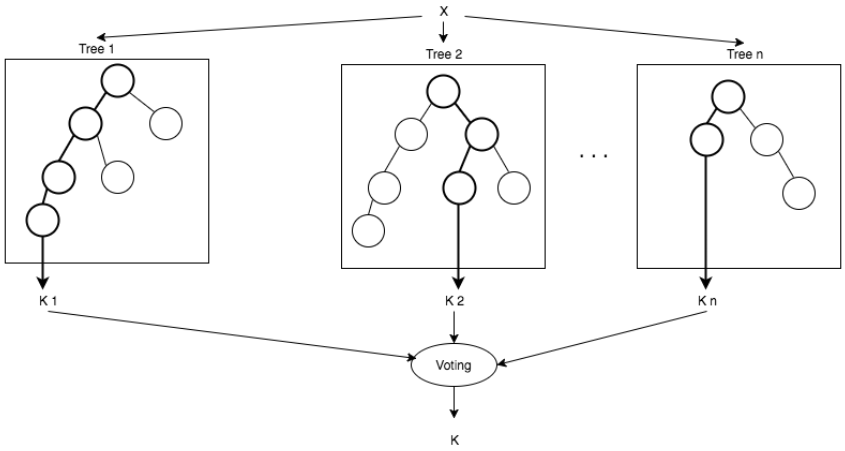
$$L2 = \sum_{i=0}^{n}(y_i - h(x_i))^2 \tag{3.3}$$

3.3 Ensemble learning methods

Ensemble learning is a set of machine learning methods that are characterized by employing multiple learning algorithms to achieve better predictions than with a stand-alone method. They are known to be one of the most successful approaches to prediction tasks, one of the reasons for this is the diversity that is curated among the ensemble members (Allende and Valle, 2017).

An ensemble learning method runs a baseline algorithm several times to form a vote based on the hypothesis. The most accurate way to do this is to construct each of the hypothesis independently if possible. The second way to construct an ensemble method is to build a connected hypothesis with weighted votes (Dietterich, 2002).

Both RF and GBT are typically used with *Classification and Regression Trees* (CARTs) as seen in figure 3.2. CARTs are decision trees used for classification and regression tasks. A decision tree is a tree-structure where each non-leafs in the tree tests some attribute, and the branches from this non-leafs represent outcomes of this test. The CARTs have different characteristics based on the method implementing them. RF uses bootstrap aggregated CARTs, while GBT uses boosted CARTs (Breiman, 2017).



The model is split into n decision trees that each vote for their proposed outcome.

Figure 3.2: Example of ensemble-learning model architecture

A machine learning model is trained until the generalization error converges. The generalization error is the prediction accuracy of machine learning models (Norvig and Russell, 2009). How well the generalization error converges is based on the number of trees, the individual tree strength and the correlation between them. Generally, many trees gives a robust forest with good prediction capabilities (Breiman, 2001).

3.3.1 Random forests

Random forests is an ensemble learning method for classification and regression, and it is built up like a forest of bootstrap aggregated CARTs where each tree depends on the values of a random vector which is sampled independently and distributed equally for all trees. A key meta-algorithm in this ensemble learning method is bagging. Bagging is used to achieve models with good fit and low variance by averaging noisy and unbiased data (Breiman, 2001). RFs are fast to train but can be slow making predictions on more extensive problems.

3.3.2 Gradient Boosting Tree

Gradient Boosting Tree (GBT) is an ensemble learning method for classification and regression. The model is built step by step and optimized by loss functions Friedman (1999). Like RF, GBT is implemented using CARTs. In this implementation type, the tree depth and type of tree differs the two methods.

GB is built up by boosted trees that in this case are shallow trees and RF is built up by bootstrap aggregated full-grown CARTs. Weak learners have high bias and low variance which means low correlation with the actual classification. Boosting is the meta-algorithm used by GB that reduces this bias. RF has the advantage of handling overfitting, while GBT has the potential to overfit.

3.4 Artificial neural networks

An *Artificial Neural Network* (ANN) is a set of computing systems loosely based on the neural networks that make up a biological brain (Gurney, 1997). In the brain, a neuron is connected to thousands of other neurons and communicate using electrical signals. When a neuron receives a signal that exceeds some threshold, it "fires" and generates a voltage pulse to relay the electrical signal.

In an ANN each neuron is represented by a node in a layered structure, with an input vector and an output vector that act as dendrites and axons. The connections between the nodes simulate synapses and the strength of the connection between them is represented by a real numbered weight (Gurney, 1997) as shown in figure 3.3.

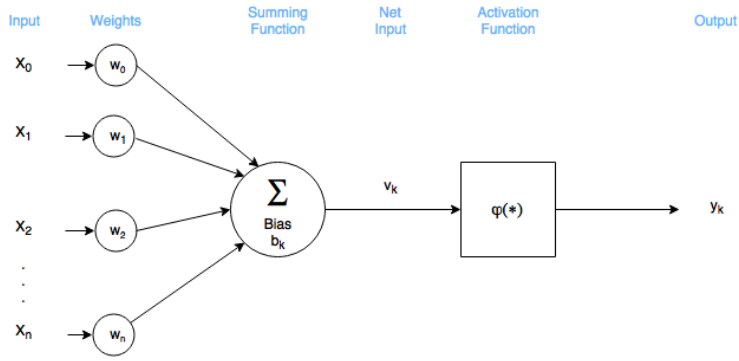


Figure 3.3: Artificial Neural Network neuron architecture

Figure 3.4 show a *Feed-forward Neural Network* (FNN) which is the simplest form of ANN. The output from each neuron is passed as an input to another neuron in the next layer. These one-way connections create a directed acyclic graph and have no internal state.

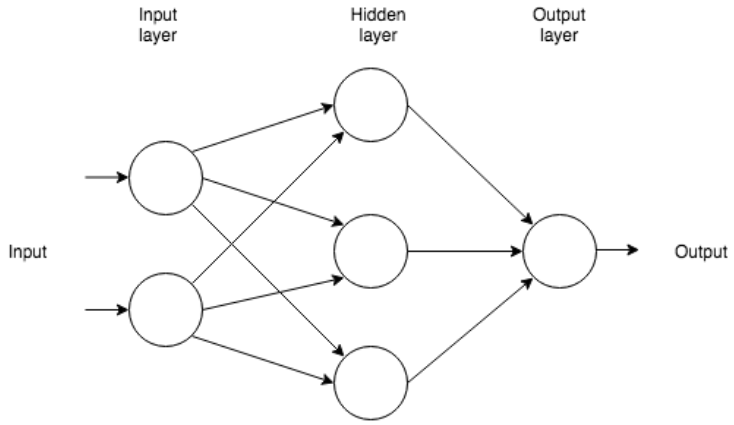


Figure 3.4: Example of Feed-Forward Neural Network model architecture

Learning with a multi-layer FNN occurs through backpropagation. For each iteration of the learning process, the observed error from the output layer is propagated back through the hidden layers. The observed error is used to calculate gradient values, which are used to update the weights between the current and previous layers. The process is repeated until all layers are updated (Norvig and Russell, 2009).

ANNs have been used to successfully solve many complex tasks, including speech recognition, machine translation, medical diagnosis and computer vision. Time series prediction is another task ANNs are commonly employed to solve. Typically, a multilayer perceptron trained with backpropagation Ortiz-Rodriguez et al. (2013) has been utilized. However, it has been shown that *Recurrent Neural Networks* (RNN) are as good or better at time series prediction due to its inherent memory mechanism.

3.4.1 Recurrent Neural Networks

Recurrent Neural Networks are a type of ANN where the output of each neuron is used as an input for the same neuron in the following step, in addition to feeding it to the next neuron in the network. This feedback-loop allows the network to keep an internal state and mimic memory (Norvig and Russell, 2009).

RNNs can model dynamic temporal behavior for a time sequence by using the internal memory, and gives it an edge by allowing an output to influence the following one. Figure 3.5 illustrates how the recurrent neural network unfolds into a complete sequence. The figure also shows how the input X_t is fed back into the network.

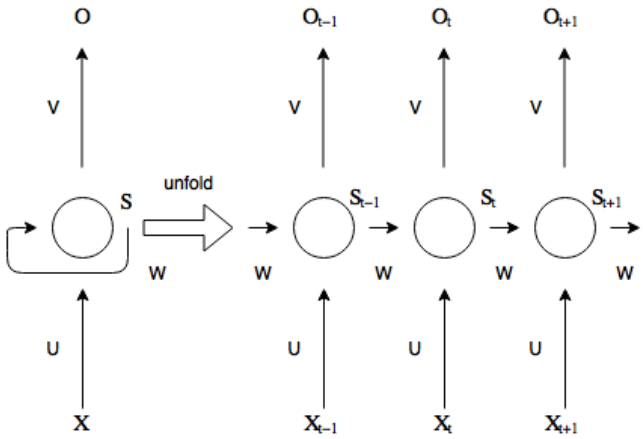


Figure 3.5: Recurrent Neural Network model architecture

3.4.1.1 Long-Short Term Memory

Long-Short Term Memory (LSTM) are memory blocks for layers in an RNN, developed by Hochreiter and Schmidhuber (1997) to deal with the long-term dependency problem, which occurs when the network fails to learn long-term connections due to another issue called the vanishing gradient problem. This is when the gradient values used to update the internal weights, vanish as they are propagated backward in the network. The vanishing gradient causes the weights in the later layers to converge before the weights in the earlier layers such that some remain largely unchanged, even though the later weights are properly fitted.

Figure 3.6 shows the architecture of the LSTM memory block and how the data flows through the gates. Information in each LSTM block is regulated through the *input gate*, *output gate* and the *forget gate* (Hochreiter and Schmidhuber, 1997). These are all controlled by sigmoid layers which output a value between zero and one depending on the learned function. The first gate the data is passed through is the forget gate. Data from the previously hidden state h_{t-1} and input data X_t is passed to the sigmoid layer which controls the amount of data to be thrown away. If the sigmoid layer outputs one, nothing is thrown away from the cell state. This output is kept in f_t to adjust the forget grade later.

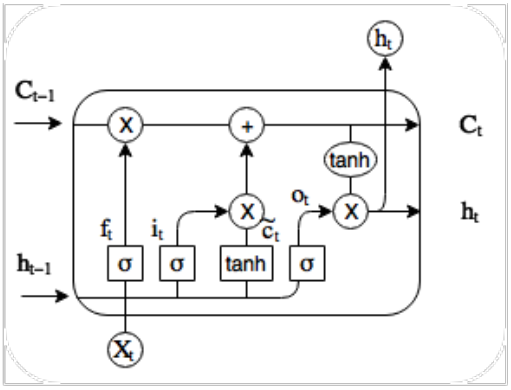


Figure 3.6: Long Short-Term Memory cell architecture

To decide what information to store in the cell state, the input data is evaluated through two different parts. The input gate which has a sigmoid layer decides which values to update and store in i_t, and a tanh layer creates new candidate vectors, $\widetilde{c}_t$. The candidate vectors are potentially new information to be stored in the cell state. A combination of the two steps creates the final update to the cell state.

After deciding what to update, the old cell state is multiplied with the forget gate f_t and added to i_t, multiplied with the candidate vectors $\widetilde{c}_t$. The last gate is the output gate which decides what values to output, o_t. The cell state is passed through a tanh layer and multiplied with the output o_t.

The advantage of this architecture over a plain RNN is the ability to let the network learn when to apply a broader context, and thus decide when to rely on long-term or short-term memory, making it possible to learn more complex functions.

3.4.1.2 Gated Recurrent Unit

Gated Recurrent Units (GRU) are a simpler and more computationally efficient version of the LSTM block (Cho et al., 2014). A GRU block contains only two gates; the input gate and the forget gate which are merged into a single update gate. The reduction in gates is what makes it more efficient than the LSTM (Cho et al., 2014). GRU take advantage of the information in the hidden states rather than using memory units to store information.

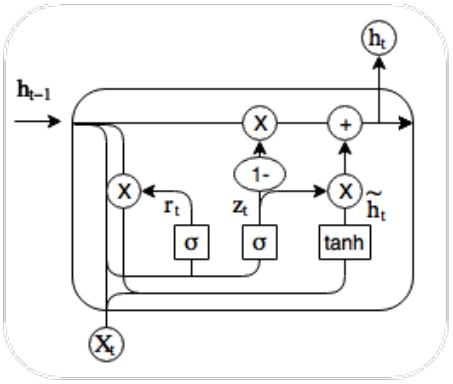


Figure 3.7: Gated Recurrent Unit cell architecture

The architecture and the two gates are shown in figure 3.7. The update gate uses a sigmoid layer and has the same task as the forget gate in the LSTM unit, that is to store information about how much previous information to keep. The information is stored in z_t as a vector. As there is no cell state in GRU, the only input is the previous hidden state h_{t-1} and current input values x_t.

The reset gate uses a sigmoid layer to determine how to combine the new input with previously memory in the hidden state. This vector is stored in r_t and will be used to update the hidden state later.

The new memory in the hidden state $\widetilde{h}_t$ is computed as a state-to-state transition with the help of a tanh layer, in the same way, candidate vectors were computed in the LSTM. The only difference is that previously hidden states are modulated through the reset gate, r_t. The output vector is computed as seen in the equation 3.4.

$$(1 - z_t) * h_{t-1} + z_t * \widetilde{h}_t \tag{3.4}$$

Cho et al. (2014) states that the GRU performance is comparable to that of the LSTM, and recent research has shown that GRUs performs better than the LSTMs on station-level demand (Chen et al., 2017).

3.5 Performance metrics

Performance metrics is an essential aspect of machine learning and selecting the right ones is crucial to being able to evaluate the results correctly. For time-series prediction problems the performance is commonly measured as the accuracy of the prediction in comparison to the real historical values.

Some of the most commonly used metrics for time-series prediction are the following:

- Root Mean Square Error (RMSE)
- Root Mean Square Logarithmic Error (RMSLE)

RMSE penalize under- and overestimates equally, while RMSLE penalizes under estimates harder. RMSE is the standard deviation of the prediction errors, and RMSLE is the square root of the squared difference between actual and predicted values, plus squared standard error (Mickey and Greenland, 1989). The standard deviation in RMSE is how many bikes the prediction error deviate from the actual demand. Both values are to be minimized. Utilizing both performance metrics helps to evaluate the models in greater depth, and provide a better basis for comparison.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.5}$$

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(a_i + 1) - log(b_i + 1))^2} \tag{3.6}$$

3.6 Validation methods

Like performance metrics, correctly validating the models is one of the most important steps in ensuring the results can be compared and evaluated. It is critical to ensure that as much of the data as possible is used for testing while keeping as much as possible for training. Probably the most important aspect when choosing validation method for a time-series problem is to ensure that the model is validated using never-before-seen data.

Some of most widely used validation methods for time-series prediction with machine learning are the following:

- K-Fold Cross-Validation
- hv-Block Cross-Validation
- Walk-Forward Validation
- 70/30 Split Validation

Walk-forward validation splits the dataset D randomly into K mutually exclusive subsets of the same size. In the first iteration, i_0 the model is trained on subset K_0 and validated on K_1. Next iteration, i_1, the model is trained on K_0 and K_1 followed by a validated on K_3.

The validation process is finished when the validation set is the last subset of the dataset. *Walk-forward validation* ensures that the model predicts on unseen data but may, however, give inaccurate results as the training set increases in size for every iteration which means that an additional error is added to the validation.

70/30 split is when the dataset divided into two parts where 70% is for training and 30% for validation. The problem with *70/30 split* is that only a small portion of the dataset is used for validation, which may give different results depending on which portion of the data is used.

K-fold cross validation avoids the drawbacks of the aforementioned validation methods by splitting the dataset such that all portions are used for validation, but can be biased by selection of the validation set on time-series problems.

A similar approach called *hv-block cross-validation* ensures that all data is used, without being biased by either the size or selection of the validation set. Using *hv-block cross-validation*, the dataset D is sliced into K subsets with same criteria as walk-forward validation. For each split s the model is trained on dataset $\hat{D}$ where D_t is removed as well as v observations from either side of D_t, and validated on D_t (Racine, 2000).

Final prediction accuracy is the mean of all validation splits. *hv-block cross-validation* ensures that the validation set is removed entirely as well as a gap of observations at either side of the validation set, to prevent that the model predicts values that have been used for training. Similar studies on the domain have shown that cross-validation results are more accurate than *70/30 split* results (Yin et al., 2012).

3.6.1 Baseline algorithm

A baseline algorithm is a basic algorithm implemented to establish a baseline performance that can be compared to the results achieved by the algorithms. If the machine learning algorithms fail to achieve higher accuracy than the baseline, there is a reason to believe that no significant patterns exist or that the models are randomly guessing.

The *persistence algorithm* is such an algorithm commonly employed as a baseline for time series prediction. The baseline is established by simply guessing that the target values stays the same as they were at the time the prediction was made. The results provide a useful reference regarding actual prediction accuracy and validation.

Chapter 4

Method

4.1 Research strategy and methodology

As the goal of this research is to evaluate machine learning methods for demand prediction in the Oslo BSS, the research is conducted as a comparative analysis of prediction algorithms. The algorithms are measured on accuracy using two measures and evaluated on prediction quality, this is accomplished through three experiments that are developed to test differences in performance and user value. The results are validated against a benchmarking algorithm and historical data.

The choice of method is based on the state-of-the-art in commonly used machine learning methods for bike sharing demand prediction, as well as what methods have been identified to suit the Oslo BSS well based on performance on the same or similar tasks. The methods that are tested are RNN LSTM, RNN GRU, RF, and GBT. The RNNs are the most promising as they have shown great potential on related tasks, but require substantially more parameter tuning than the RF and GBT algorithms, which may make it harder to achieve a similar level of performance using the Oslo BSS data.

The models produced by these methods are validated using *hv-block cross-validation* with ten splits, in addition to a persistence baseline algorithm and a visual inspection of each prediction graph.

4.2 Feature selection

The features utilized in the experiments are selected through a process of data analysis and model optimization. Features that show promise in related work are examined to determine if they are applicable in the Oslo BSS. They are analyzed to determine correlations and tested in the models to reveal if they yield accuracy improvements. The selected features are listed in section 5.3.1.

4.3 Station selection

A selection of stations is used to reduce the complexity of the experiments and implementation while maintaining a dataset that represents the variations in the BSS.

Each experiment is run with a set of stations that are selected based on the dominance of the demand, the traffic volume and the duration of the operation. The goal is to select a set of stations representing the most significant variations, ensuring that the data gathered for these stations represent the whole BSS.

The type of dominance a station exhibit is determined by the ratio between departures and arrivals. A station with many more departures may be more frequently empty than a station with an approximately equal traffic volume. When a station is empty, it interrupts the demand pattern, making it inconsistent and harder to predict. Stations with either 20% more arrivals than departures, or 20% more departures than arrivals are classified as dominant.

The volume of traffic varies significantly between stations and can result in very different demand profiles. To account for the variety, the selected stations have all been operational since the opening in 2016 and spans a large part of the range in traffic volume.

1. **Departure dominant** $> 20\%$ departures
2. **Neutral** $\sim 0\%$ difference between arrivals and departures
3. **Arrival dominant** $> 20\%$ arrivals

The selected stations are listed in section 5.3.2.

4.4 Data preparation

Data preparation is an essential and often overlooked part of predictive modeling. Properly prepared data can make the difference between achieving excellent results and achieving no results and can have a significant impact on learning time and end performance. (Yu et al., 2006). Although the data utilized in this thesis is of good quality, it must be transformed, cleaned and scaled to enable the algorithms to learn the hidden patterns properly. The methods employed are not necessarily specific to RNNs or ensemble methods, and can be applied to many kinds of algorithmic data analysis. Table 4.1 shows an example of raw data fully prepared for each method.

	Raw data	RNN	Ensemble learning methods
Started trips	21	0.75	21
Ended trips	3	0.1	3
Hour	8	One-hot-encoded	8
Temperature	14.0	0.6	14.0
Rain	0.0	0.0	0.0
Available bikes	4	0.1	4
Weekday	Monday	1	1
Sun minutes	60.0	1.0	60.0
Wind speed	4.3	0.8	4.3
Minutes	60	x	x

Table 4.1: Dataset pre-processing example

Data variable selection is the first step in the process of preparing data for prediction. As the datasets are stored on the GCP BigQuery service, data for each station can easily be extracted using standard SQL queries. Preliminary filtering using SQL is done to avoid superfluous loading data such as ids and other columns that can not be used for training.

The *trips* table is filtered to remove erroneous or invalid trips by discarding trips that have a duration of under 60 seconds and that are not registered as *delivered* or *cancelled*. These entries do not represent the normal usage-patterns of the system and must be removed to reduce noise. Timestamps that are stored in Coordinated Universal Time are converted to Central European Time such that the data can be merged on the correct timestamp later in the process. The datasets are indexed on their respective timestamp. All tables are re-sampled to a common period such that the models can be trained for specific prediction spans.

4.4.1 Derived features

The demand estimates are derived from the trip records and provided as an estimate for the real demand. It is calculated by taking the arrival and departure-rate of bikes within the period there are bikes available in the station, and extrapolating this for when the demand is censored due to depletion.

If the next period is empty, the departure-rate is assumed to remain unchanged and is forward filled using the preceding rate. As this formula can result in unrealistic values, the maximum demand is limited to three times the size of the station. This figure was established in cooperation with the system operators, and was deemed to represent a reasonable maximum demand for a station in the given period.

4.4.2 Algorithm specific preparation

The features vary widely in both format and scale and must be normalized such that the RNNs can consider each feature equally, and to avoid large scaled values overshadowing others(Norvig and Russell, 2009). This is accomplished by scaling each variable to a value between one and zero based on the minimum and maximum value observed in each column.

For the algorithms to be able to utilize timestamps, they must be decomposed into specific features representing interesting aspects of the timestamp, such as the month, hour, and day of the week. This is sufficient for RF and GBT, but for the RNNs to take advantage of this, they must be further decomposed. Here, one-hot encoding has been utilized. One-hot encoding separates continuous values into several features for each possible value. The value of these features are either one or zero to indicate if the value is present, an example of this is shown in table 4.2 where the values indicate the month of May.

Month_5	Month_6	Month_7	Month_8	Month_9	Month_10	Month_11
1	0	0	0	0	0	0

Table 4.2: One-hot encoding example

To use the datasets for supervised learning with RF and GBT the target feature is shifted one period backward so that the input for time t appears with the target value for time $t + 1$. A major benefit of the RNN is to utilize several periods of data in one prediction easily. To prepare the data for this, the variables are shifted backward one period. In addition, sets of variables from the n preceding periods are appended. This makes it possible to predict time t using features from $t - 1$ to $t - n$.

4.4.3 Cleaning and data removal

The final step in the data-preparation process is cleaning and data removal. Illogical and missing values can have a detrimental effect on the learning ability of the system as these deviate from the underlying patterns in the data. The easiest way to remedy this is merely to remove the erroneous rows. Data that is logged while the system is closed at night or for the winter is also considered to be invalid and thus deleted.

4.5 Architecture

The general support architecture, described in figure 4.1, shows the complete process from raw data to final prediction.

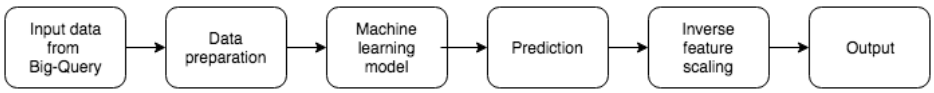


Figure 4.1: General machine learning support architecture

4.5.1 RNN structure

Figure 4.2 shows the architecture of the two RNN versions. Both are based on Keras-models implemented in Python. The first layer consists of neurons where each represents an input feature. The hidden layer consists of 100 memory blocks. The third layer is a *dropout layer* with a dropout probability of 10%.

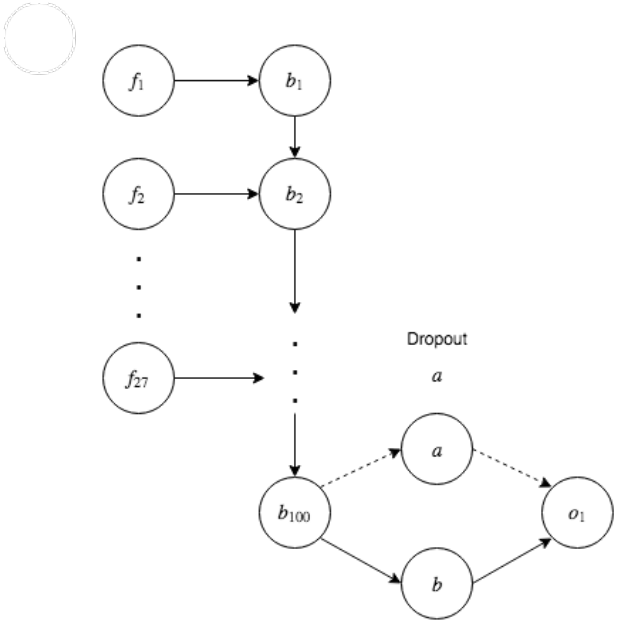


Figure 4.2: Implemented architecture for RNNs

Of the two RNNs, one implement LSTM blocks, and the other implement GRU blocks. Each of these consists of several cells where each cell represents a memory for a specific time-step in the time-series data. All the cells in one block are called a window. The size of the window is tuned for each problem.

4.5.2 RF and GBT structure

The RF and the GBT are built up by 50 estimators and a prediction class as shown in figure 4.3. The class makes a prediction based on votes from each tree in the forest, weighted by their probability estimates. The class with the highest mean probability estimate across the trees dispose of the final prediction. RF and the GBT were implemented in Python using the Scikit-learn library.

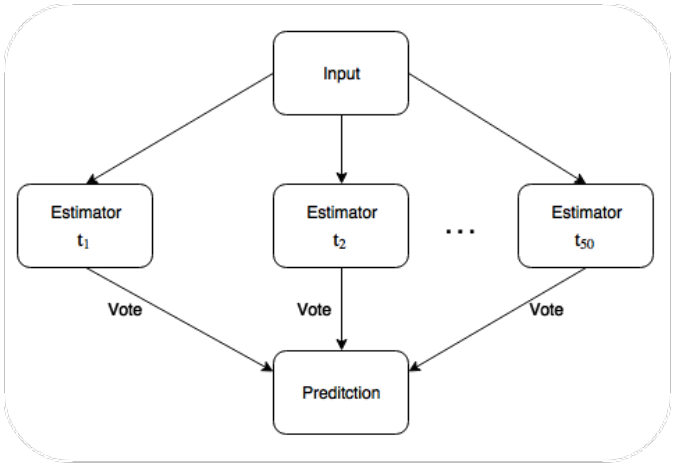


Figure 4.3: Implemented architecture for ensemble learning models

4.6 Tools

The following are tools needed for the implementation.

Google Cloud Platform

Google Cloud Platform (GCP) [1] is a cloud-based platform for storing data, hosting and building applications, machine learning, and management. GCP uses the same scalable infrastructure Google use for products like Google search and YouTube.

UIPs systems are mainly based on GCP, so compatibility is an important aspect that was considered when selecting tools. Using tools that are compatible makes it easier to extend the models such that they can be implemented and used in production environments. An introduction to GCP was provided by Google at their office in Oslo to kick-start the development process.

Google BigQuery

BigQuery [2] is an SQL database service optimized for big data analysis and handling. Processing large datasets can be a bottleneck in machine learning. This is mitigated by utilizing this service to run queries on Google servers with the amount of CPU's necessary, in order to resolve long queries in a matter of seconds. The processing power helps maintain a stable and fast data-flow and ensures that data can be updated and modified easily.

[1] https://www.cloud.google.com/
[2] https://www.cloud.google.com/bigquery/

TensorFlow

TensorFlow [3] is an open-source python library for numerical computation, developed for machine intelligence research. This library works perfect for building models in Google Machine Learning Engine (ML-engine) and is also used in internal Google products like Photos and Cloud Speech. Like BigQuery, this library works with large data and data of any type. Tensorflow was chosen based on compatibility with ML-engine and a good graphical user interface called Tensorboard. This provided good analysis tools to observe training and tune parameters of each model.

Keras

Keras [4] is an open-source neural network library that can be accessed as a high-level interface. It is written in Python and is therefore capable of running on top of TensorFlow. It was chosen because it enables easy experimentation with different neural network models and comparison of performance without spending excessive time on implementation.

Scikit-learn

Scikit-learn [5] is a machine learning library written in Python used for classification, regression and clustering problems. Scikit-learn was used to implement RF and GBT.

Pandas

Pandas [6] is an open source python library used for data structures and data analysis. This library provided every model with a Two-dimensional size-mutable data structure called DataFrame. Pandas were chosen based on high performance and the many available data manipulation methods.

[3] https://www.tensorflow.org/
[4] https://www.keras.io/
[5] http://www.scikit-learn.org/
[6] https://www.pandas.pydata.org/

Chapter 5

Experiments

5.1 Approach

The process of training is slightly different for the individual methods. For each experiment, the RNN and GBT are trained until the loss converges. The RF is built using the entire training set. The models are validated using *hv-block cross-validation* with ten splits, in which the models are trained using 85% of the full dataset and validated using the remaining 10%. 5% is left out to avoid overlaps. The resulting accuracy is the mean of these 10-splits.

5.2 Design

5.2.0.1 Experiment overview

The three experiments shown in table 5.1, are designed to resolve the second and third research questions. These are crafted to test performance in a wide range of configurations and provide a solid basis for comparing the algorithms. The first compares a 30-minute prediction-span to a 60-minute span. The second examines the difference in training with a full versus half dataset, and the last is designed to find the difference in performance between estimated demand and censored demand.

ID	RQ	Description
E1	RQ2/RQ3	Compare performance of 30min and 60min timeframe
E2	RQ2/RQ3	Compare performance of complete and limited dataset
E3	RQ2/RQ3	Compare performance of censored demand and estimated demand dataset

Table 5.1: Experiment matrix

The training time for machine learning methods is often an essential aspect of evaluation, as this affects the practicality and viability of the models. In this case, the models are planned to be deployed to google cloud platform, which provides scalable computing power, to enable the training to be completed almost as fast as wanted. In addition, the system is closed during the night so that any model updates can be completed during this period. Due to this the training time aspect of performance is not tested.

5.2.1 E1: Prediction-span size

The prediction-span is the range in time forward for which the target value is aggregated. In this problem, the target value is the number of started trips from each station within the specified interval. For the estimated demand dataset this also includes the number of people who would have biked from it had it not been empty.

The prediction-span is a vital parameter to examine, as the target values fluctuate significantly and a change in the length of the prediction-span significantly alters the predictability and slope of the demand curve. Changing the size of the prediction-span can also yield different values. A too short outlook may be more accurate but might not allow the users to react to it, while a larger value may provide inadequate detail.

In this experiment all models are trained on two datasets, one sampled to 60 minutes, predicting 60 minutes forward, and one sampled to 30 minutes, predicting 30 minutes forward. These values are selected because they have proven to yield good results in similar experiments (Yin et al., 2012; Chen et al., 2017) and provides the most value to the stakeholders in the BSS. This shows if any of the methods may be more suitable for either prediction-span and if the more detailed 30-minute data yield a better result than the 60-minute data.

5.2.2 E2: Dataset size

One of the most crucial aspects of machine learning is the amount of data available. Often more data yield better results, but more data can also decrease quality if the patterns in the data changed. In this case, the data spans two consecutive seasons, in which the system has been continuously expanded with more bikes and more stations, and the number of trips within the system has increased by more than half a million. This expansion also means that there exist numerous stations that have less than a year of recorded data. All of this makes experiment two important to find out which algorithm performs best on stations with a limited dataset, and which can best take advantage of the full dataset.

To accomplish this, only stations that have been available since April 2016 have been selected for testing across all experiments. All models are first trained on the complete dataset spanning both 2016 and 2017, and then only in 2017. This eliminates inconsistencies between stations and shows what difference can be found between these two datasets. The estimated demand dataset, however, only includes data for 2017, so to yield similar results this dataset is approximately cut in half such that the models first are trained using data for the whole of 2017 and then from mid-summer until the end of the season in November.

5.2.3 E3: Estimated demand

This experiment provides the basis for comparing the accuracy between estimated demand and the censored demand datasets. Even though the estimated demand dataset is based on the censored demand dataset, there may be pattern differences that the algorithms handle differently.

Predicting the estimated demand is interesting as this allows the system operators to know not only the number of people who rent a bike, but also who would like to but are unable to due to depletion. The predictions can help the operators adjust the number of bikes and slots at each station as well as rebalance the stations with an optimal number of bikes.

The estimated demand data is not validated. Reviewing the data reveal values that with high probability can be classified as incorrect, so there is a reason to believe that this affects the results.

5.3 Setup

The baseline settings for each algorithm is based on previous research, common knowledge about machine learning architecture as well as trial and error optimization. Each algorithm was tested with a different number of hidden layers, estimators, blocks, and cells within a frame that is reasonable for the size and type of problem. The parameters are kept constant across all station to ensure that a common model that fits all stations is produced.

5.3.1 Features

The following features are utilized in the prediction models with a few variations listed in parentheses.

Started trips
: The number of trips started from the current station during the current period

Ended trips
: The number of ended trips at the current station during the current period

Available bikes
: The number of available bikes at the end of the period

Minute (30 min prediction only)
: The minute of the hour

Hour
: The hour of day

Weekday
: The day of the week

Rain
: Hourly amount of precipitation in millimeters

Temperature
: Hourly average temperature in Celsius

Sun minutes (GBT and RF only)
: Minutes of sun during the current period

Wind speed (GBT and RF only)
: Hourly average wind speed

5.3.2 Stations

Table 5.2 show the selected stations. The absolute difference is the difference in the total amount of started and ended trips spanning the entire duration the station has been open, and is what determines the station profiles.

Station ID	Station profile	Name	Trip count	Absolute difference
279	Arrival	Bankplassen	24 488	22 884
234	Arrival	Spikersuppa Vest	37 639	33 866
204	Arrival	Paléhaven	55 133	24 832
277	Arrival	Helga Helgesens plass	81 098	10 416
253	Arrival	Aker Brygge	99 541	35 442
291	Departure	St. Hanshaugen park nord	30 101	18 184
183	Departure	Storo Storsenter	32 159	19 162
161	Departure	St. Hanshaugen park vest	39 775	21 124
233	Departure	Alexander Kiellands Plass	131 173	17 293
267	Departure	Bislett Stadion	116 229	21 870
190	Neutral	Parkveien	55 466	10
245	Neutral	Kværnerbyen	30 073	135
210	Neutral	Birkelunden	33 885	263
251	Neutral	Hallénparken	70 874	582
191	Neutral	Jacob kirke	80 769	513

Table 5.2: Experiment stations overview

5.3.3 RF setup

The number of estimators is a critical element of tuning the RF. For the Oslo BSS, 50 estimators were found to produce the best predictions. The estimators are unpruned meaning that the trees are uncapped and variance is reduced. When using the datasets provided for this research, there are no issues with memory consumption and complexity. MSE is used as a split quality measure for training the RF.

Key parameter: *50 estimators*

5.3.4 GBT setup

The baseline for training GBT is established with *least squares* as loss function and *Friedman MSE* function for split quality measure. A max tree depth of five was found to produce the best quality predictions, with a learning rate of 0.1.

α = learning rate
δ = max tree depth

Key parameter: *50 estimators,* $\alpha = 0.1, \delta = 5$

5.3.5 RNN setup

Both RNN models are initialized with *Glorot uniform initializer* which was found to bring the model to convergence quickly (Glorot and Bengio, 2010).

One of most significant parameters for the RNN was the window size. The best was found to be seven steps, which means the memory block utilized seven time-steps of previous data to produce prediction t.

Due to the difference in training data available in the estimated- and censored demand datasets, more training was needed for convergence on the censored demand dataset. The convergence was achieved by using a larger number of training epochs.

Varying the optimizer algorithm did not alter the performance significantly. *Adam-, Adadelta-* and *Adagrad optimizers* had similar results in terms of learning, but Adagrad was selected.

To not penalize the outliers in training, MAE was used as loss function. Both RNNs were trained with batches of size 32.

α = learning rate
λ = loss function
γ = Activation function
ι = kernel initializer

5.3.5.1 RNN - estimated demand dataset

Key parameters: *100 neurons, 50 epochs,* $\alpha = 0.01, \lambda = mae, \gamma = adam, \iota = glorot - uniform$

5.3.5.2 RNN - censored demand dataset

Key parameters: *100 neurons, 100 epochs,* $\alpha = 0.01, \lambda = mae, \gamma = adam, \iota = glorot - uniform$

Chapter 6

Results and evaluation

6.1 Experimental results

The complete results of experiment E1, E2 and E3 are shown in table 6.1 and figures 6.1, 6.2 and 6.3. These show the mean accuracy of the machine learning models across all stations, measured with RMSE and RMSLE respectively. They are compared across different prediction-spans, datasets, and dataset sizes. Tables 6.2, 6.4 and 6.6 highlight the model with the highest accuracy on each set of variables within the specified dataset. The results represent the predictive score of each model, achieved by training the RNNs, GBT, and RF individually for each set of variables.

In general, all models achieved good prediction accuracy on the censored demand dataset. The LSTM RNN model was found to outperform the other models generally and is used to exemplify the difference in the experiments and the quality of the predictions. Station 191, *Jakob kirke*, is used to exemplify the difference in prediction quality as it represents the overall results well. It is a neutral station with a high traffic volume of 80 thousand trips, situated about midway between the center and the outskirts of the BSS. A period of high traffic volume is selected to amplify features in the demand profiles and provide a varied basis for comparison.

	60 Minutes		30 Minutes	
	Complete	**Limited**	**Complete**	**Limited**
RNN LSTM	**3.993**	**4.109**	**2.557**	2.597
RNN GRU	4.139	4.151	2.622	2.637
RF	4.163	4.151	2.675	2.653
GBT	4.251	4.116	2.589	**2.561**
Baseline	5.618	5.810	3.359	3.461

(a) RMSE

	60 Minutes		30 Minutes	
	Complete	**Limited**	**Complete**	**Limited**
RNN LSTM	**0.575**	**0.583**	**0.551**	**0.557**
RNN GRU	0.597	0.595	0.564	0.568
RF	0.610	0.590	0.603	0.600
GBT	0.633	0.610	0.585	0.578
Baseline	0.774	0.788	0.714	0.728

(b) RMSLE

Table 6.1: Experiment results overview
The best score in each column is highlighted in bold

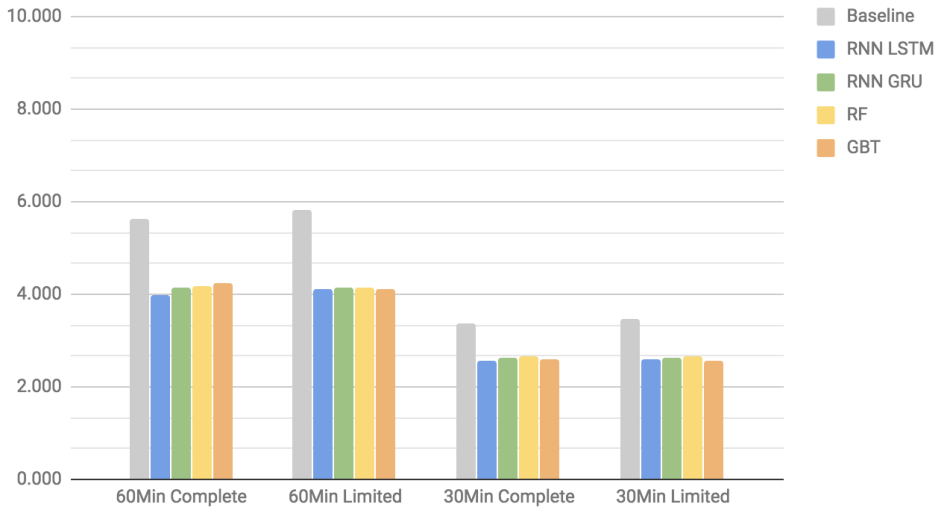


Figure 6.1: Experiment results censored demand (RMSE)

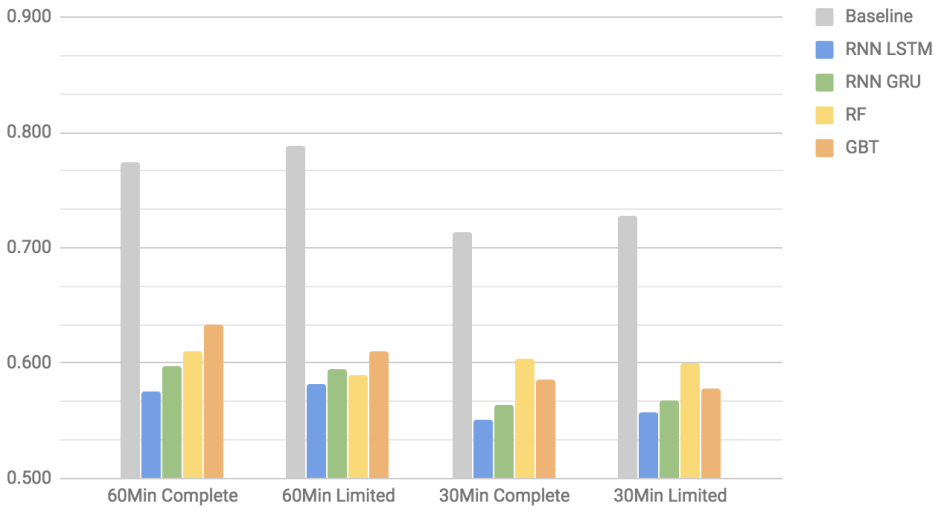
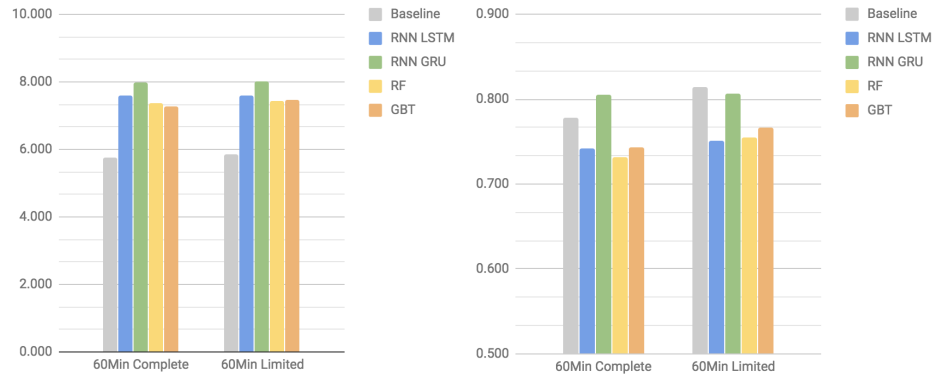


Figure 6.2: Experiment results censored demand (RMSLE)



(a) RMSE **(b)** RMSLE

Figure 6.3: Experiment results estimated demand (RMSE and RMSLE)

6.1.1 E1: Prediction-span size

The purpose of E1 is to discover differences in how the models handle the two prediction-span sizes. Table 6.2 shows that the LSTM RNN consistently outperform the other models on both the 30- and 60-minute prediction-span, followed by the GRU RNN. Among the ensemble models, the RMSLE shows the RF performs best with the 30-minute prediction-span, while the GBT performs best with the 60-minute span. All models far exceed baseline.

	Mean RMSE		**Mean RMSLE**	
	60 minutes	**30 minutes**	**60 minutes**	**30 minutes**
RNN LSTM	**3.993**	**2.557**	**0.575**	**0.551**
RNN GRU	4.139	2.622	0.597	0.564
RF	4.163	2.675	0.610	0.603
GBT	4.251	2.589	0.633	0.585
Baseline	5.618	3.359	0.774	0.714

(a) Complete dataset

	Mean RMSE		**Mean RMSLE**	
	60 minutes	**30 minutes**	**60 minutes**	**30 minutes**
RNN LSTM	**4.109**	2.597	**0.583**	**0.557**
RNN GRU	4.151	2.637	0.595	0.568
RF	4.151	2.653	0.590	0.600
GBT	4.116	**2.561**	0.610	0.578
Baseline	5.810	3.461	0.788	0.728

(b) Limited dataset

Table 6.2: Experiment 1 results
The best score in each column is highlighted in bold

Comparing the accuracy with the 30- and 60-minute prediction-span yields surprising results. Considering that an increased look-ahead generally increases uncertainty and that the 30-minute dataset is double the size of the 60-minute dataset as a result of the higher sample rate, one would expect the models using 30-minute prediction-span to far exceed the accuracy achieved with the 60-minute prediction-span, which is not the case. With the full dataset, the difference in RMSLE between the two spans for the RNN LSTM, RNN GRU, and GBT is small, and there is no difference in accuracy for the RF. The baseline improved the most as shown in table 6.3.

	RMSE	**RMSLE**
RNN LSTM	36%	4%
RNN GRU	37%	5%
RF	36%	0%
GBT	38%	6%
Baseline	40%	8%

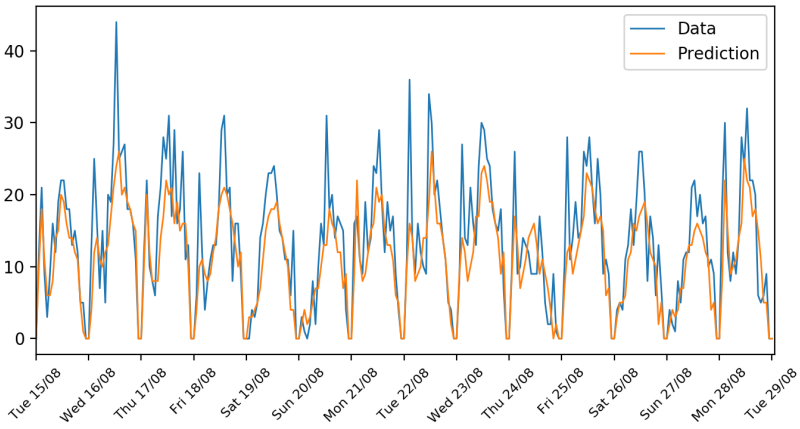
Table 6.3: Overview of accuracy change E1

A likely cause of the negligible difference may be that an improvement in prediction accuracy induced by the bigger dataset is negated by the increased volatility caused by a higher sample rate. This increase in volatility is present in figure 6.4.

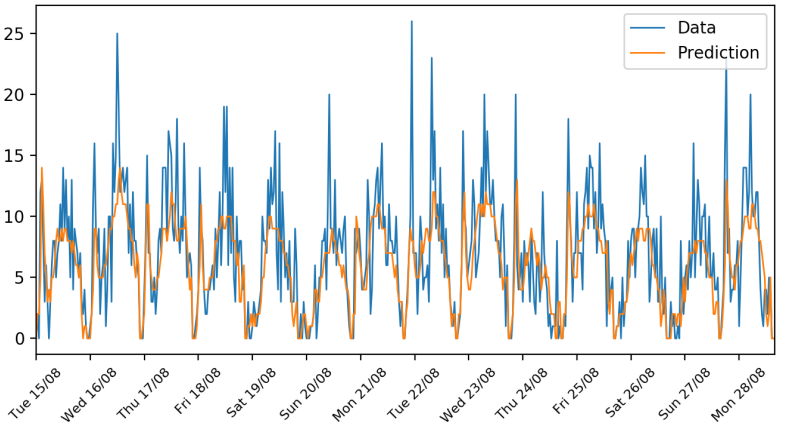
For all models the 30-minute prediction-span shows a mean improvement of 36% RMSE, which can be expected as the reduction from 60- to 30-minute spans lowers the mean of the dataset, reducing the scale of the RMSE. This also affects the RMSLE but to a lesser degree. Examining figure 6.4 shows this reduction and a significant difference in the size of the demand peaks.

The small difference in RMSLE indicates that the factor with which the prediction was off is nearly the same. The fact that the baseline improved the most shows that the models performed worse compared to the baseline on the 30-minute dataset.

Figure 6.4 also shows that the models trained on the 30-minute dataset seem to predict the sharp morning peaks better, while the models trained on the 60-minute dataset is better at predicting the afternoon spikes.



(a) 60 minutes station 191 LSTM complete



(b) 30 minutes station 191 LSTM complete

There are a number of important differences in the prediction graphs. The pattern of the 30 minute data (b) is more volatile than the 60 minute data (a). The scale of (a) is almost twice that of (b). However, both graphs follows the same general pattern and generally predict lower demand.

Figure 6.4: Comparison of prediction quality 30- and 60 minutes

6.1.2 E2: Dataset size

The goal of E2 is to discover how the models handle different dataset sizes. The results are shown in table 6.4 and shows that the LSTM RNN perform better than the other models on both the complete and limited dataset. Comparing the mean across all models show no significant difference concerning prediction accuracy. The GBT and RF achieve a slightly better accuracy on the limited dataset, while the RNN LSTM perform slightly worse. The RNN GRU did not change significantly. The results indicate that the LSTM RNN takes advantage of the full dataset, while the ensemble methods become less accurate. All models far exceed baseline accuracy with both the complete and limited dataset. The trend is the comparable for both the 30- and 60-minute prediction-spans.

	Mean RMSE		**Mean RMSLE**	
	Complete	**Limited**	**Complete**	**Limited**
RNN LSTM	**3.993**	**4.109**	**0.575**	**0.583**
RNN GRU	4.139	4.151	0.597	0.595
RF	4.163	4.151	0.610	0.590
GBT	4.251	4.116	0.633	0.610
Baseline	5.618	5.810	0.774	0.788

(a) 60 minutes

	Mean RMSE		**Mean RMSLE**	
	Complete	**Limited**	**Complete**	**Limited**
RNN LSTM	**2.557**	2.597	**0.551**	**0.557**
RNN GRU	2.622	2.637	0.564	0.568
RF	2.675	2.653	0.603	0.600
GBT	2.589	**2.561**	0.585	0.578
Baseline	3.359	3.461	0.714	0.728

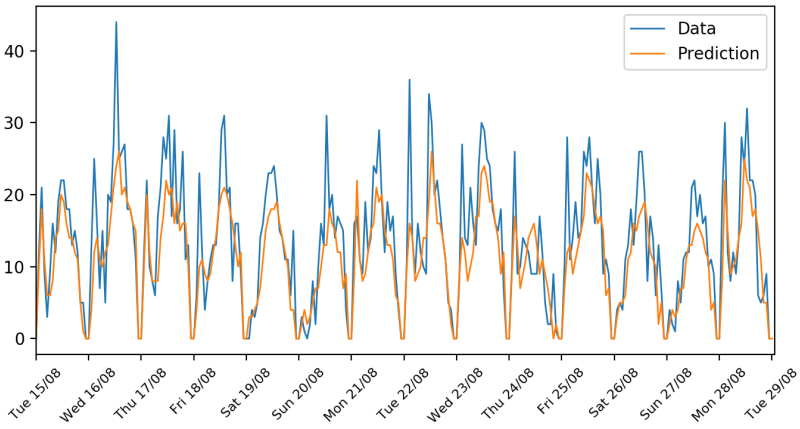
(b) 30 minutes

Table 6.4: Experiment 2 results
The best score in each column is highlighted in bold

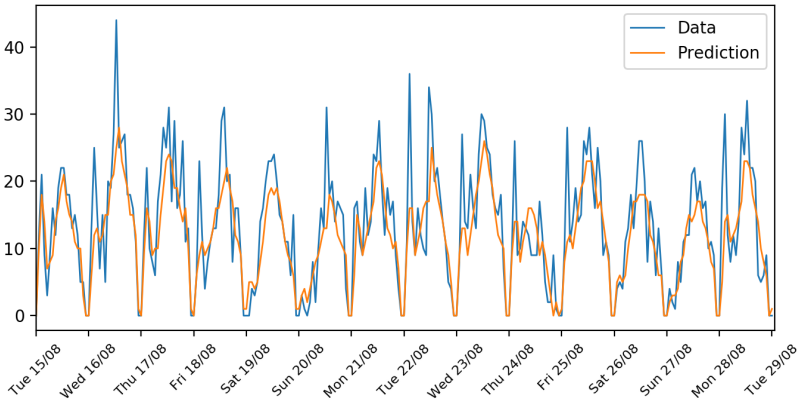
Figure 6.5 shows that the LSTM RNN is better at predicting the morning demand peaks on the complete dataset compared to the limited dataset. The opposite is observed for the ensemble models. This observation is supported by the change in accuracy seen in table 6.5.

	RMSE	**RMSLE**
RNN LSTM	-2%	-2%
RNN GRU	0%	0%
RF	1%	4%
GBT	2%	5%
Baseline	-3%	-4%

Table 6.5: Overview of accuracy change E2



(a) Complete dataset station 191 LSTM 60 minutes



(b) Limited dataset station 191 LSTM 60 minutes

There are few visible differences in prediction quality among the complete and limited dataset. Models trained with the complete dataset (a) are better at predicting the morning spikes. This is visible when comparing Thursday 17. and Monday 28. where the predictions for the morning peaks are absent in (b).

Figure 6.5: Comparison of prediction quality limited and complete dataset

6.1.3 E3: Estimated demand

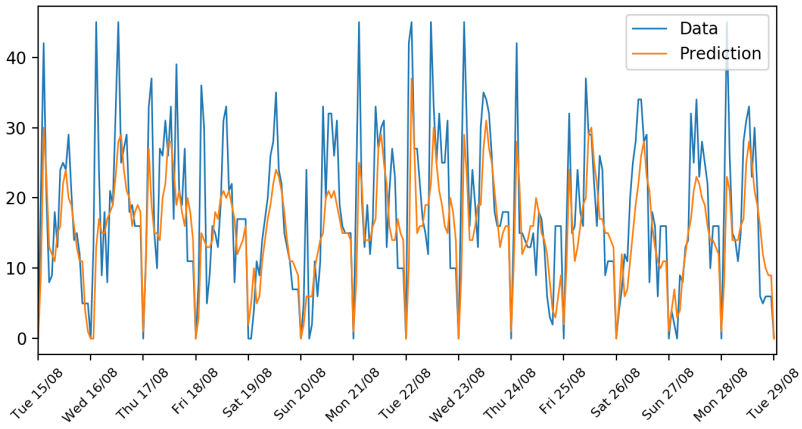
The purpose of E3 is to determine if the models are able to predict estimated demand, and show the difference in how they perform. Table 6.6 shows that all models consistently perform significantly better with the censored demand than with the estimated demand data. For the estimated demand dataset the ensemble models performed best, with the GBT achieving the best RMSE and the RF achieving the best RMSLE.

	Mean RMSE		**Mean RMSLE**	
	Estimated	**Censored**	**Estimated**	**Censored**
RNN LSTM	7.590	**3.993**	0.743	**0.575**
RNN GRU	7.994	4.139	0.806	0.597
RF	7.373	4.163	**0.731**	0.610
GBT	**7.266**	4.251	0.743	0.633
Baseline	5.745	5.618	0.778	0.774

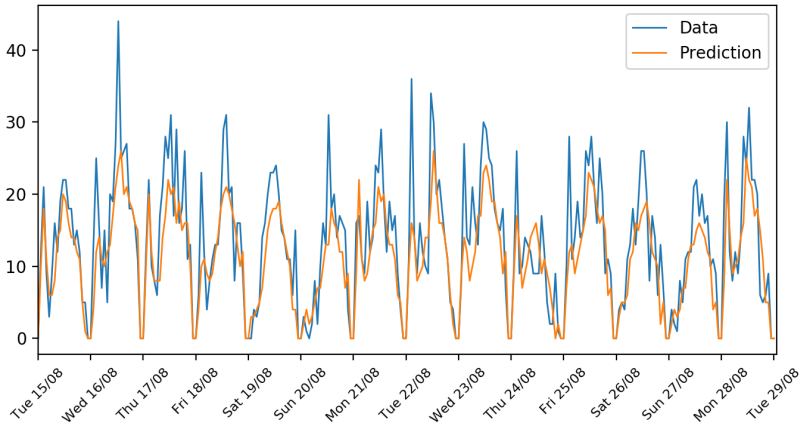
Table 6.6: Experiment 3 results

However, compared to the baseline prediction, all models achieved a considerably worse RMSE and a slightly better RMSLE. Failing to beat the baseline prediction indicates that the models are unable to learn any complex patterns or that there exists none in the data.

Despite the baseline being nearly identical for both the censored- and the estimated datasets, indicating that the variance is comparable, the estimated demand was found to have a more noisy and erratic pattern than the censored demand during data analysis. The fact that the RMSE is significantly worse while the RMSLE is slightly better indicates that the spikes are more random in size and occurrence, and is likely part of what is causing the difference in performance. This can be observed as sudden sharp spikes in figure 6.6a.



(a) Estimated demand station 191 LSTM 60 minutes complete



(b) Censored demand station 191 LSTM 60 minutes complete

The graphs depict the same period for the same station. The spikes in the estimated demand data (a) are generally much higher than in the censored demand data (b). However, the patterns in both graphs are similar. Though, the estimated demand predictions are punished harder, as they deviate much more from the target values, due to the more random pattern.

Figure 6.6: Comparison of prediction quality estimated- and censored demand

6.2 Prediction accuracy summary

Table 6.7 shows the highest accuracy achieved by each model in addition to which parameters this was accomplished with.

	Best mean RMSLE	Timeframe	Dataset size
RNN LSTM	0.551	30 Minutes	Complete
RNN GRU	0.564	30 Minutes	Complete
RF	0.590	60 Minutes	Limited
GBT	0.578	30 Minutes	Limited

(a) RMSLE

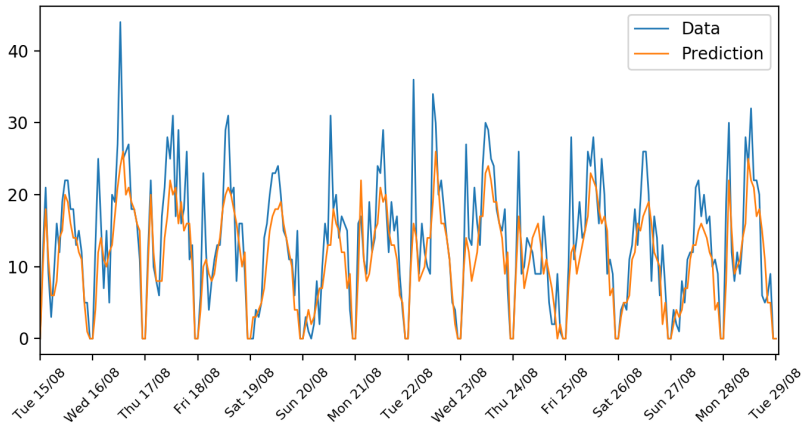
	Best mean RMSE	Timeframe	Dataset size
RNN LSTM	2.557	30 Minutes	Complete
RNN GRU	2.622	30 Minutes	Complete
RF	2.653	30 Minutes	Limited
GBT	2.561	30 Minutes	Limited

(b) RMSE

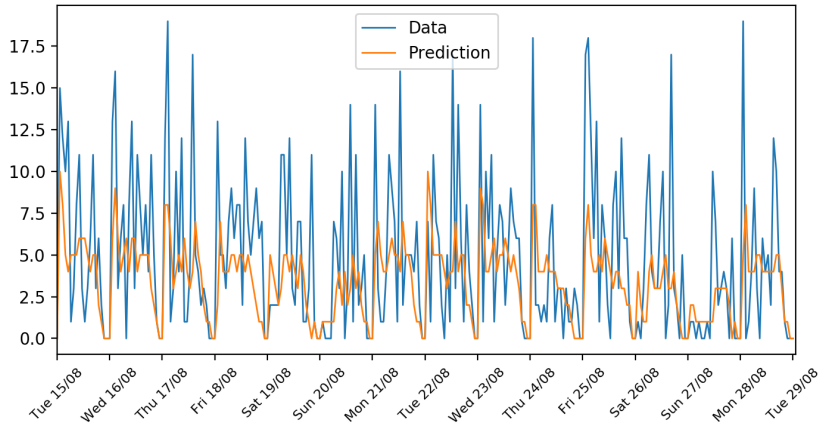
Table 6.7: Highest accuracy with corresponding parameters

6.3 Evaluation and analysis

The quality of predictions was found to vary depending on station and traffic volume. Figure 6.7 shows this. The most likely reason for the variation is the predictability of the patterns due to the profile of the stations. Figure 6.9 and 6.12 show that the models often underestimate the demand peaks, which can be considered to be normal when the data fluctuates the way the demand data does. It can be a result of a low mean target value in the dataset, which makes predicting sudden spikes that deviate from the mean challenging.



(a) Station 191 Censored demand RNN LSTM 60 minutes Complete



(b) Station 291 Censored demand RNN LSTM 60 minutes Complete

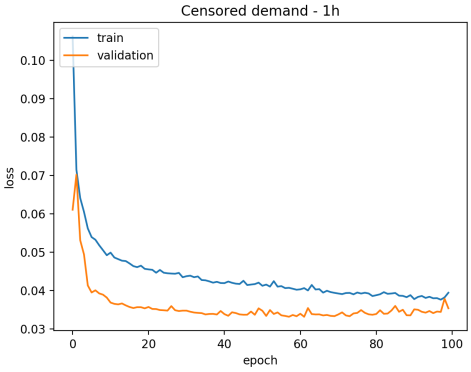
The difference in prediction quality spans a wide range. These stations have been trained using the same model and parameters but has a very different prediction quality.

Figure 6.7: Comparison of station prediction quality

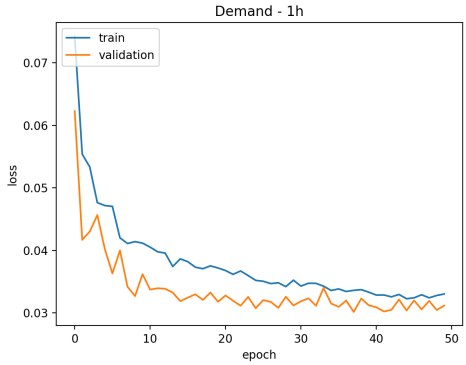
Among the two variants of the RNNs, there are only slight differences in performance. The LSTM RNN generally performed best of the two, with the GRU RNN following closely on all experiments. The two blocks are closely related and should provide similar results. The slight difference may be explained by the fact that the LSTM units are theoretically better at handling more extended sequences (Hochreiter and Schmidhuber, 1997). The most significant difference was observed with the estimated demand dataset. However, this can most likely be attributed to the general performance reduction induced by increased volatility in this dataset.

Figure 6.8 shows the loss for the RNNs on the estimated and censored demand dataset. All graphs show that the models converge and that they have been sufficiently trained. This is evident as a set of graphs that slowly flatten out and meet in the end. At the start of both graphs, the validation error is lower than training error. The RNNs includes a dropout layer to prevent overfitting in training and ensure later convergence, which is why the validation error is lower than the training error (Srivastava et al., 2014). This is because the validation is run without dropout.

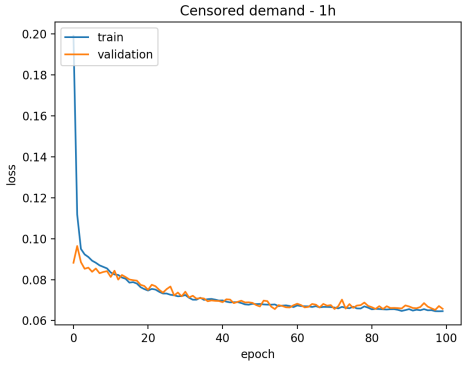
Figure 6.9 shows that the RNNs manages to predict the occurrence of demand spikes relatively well, but struggle to predict the volume accurately. Predicting the size of these spikes is a difficult task, as they can change depending on irregular factors such as manual bike relocation and censoring. In the figure the size of the demand spikes at Wednesday 16/8 and Tuesday, 22/8 can likely be attributed to manual relocation. The LSTM is slightly better than the GRU at estimating, but both generally underestimate the demand. The LSTM is also better at predicting the occurrence of the morning demand peaks.



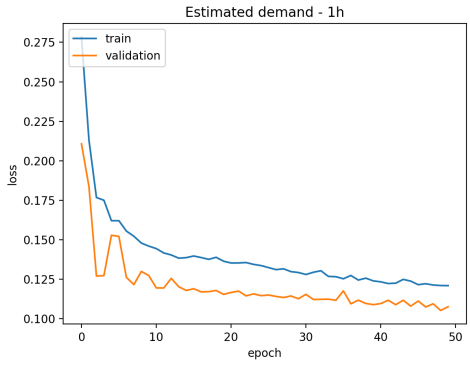
(a) Censored demand 60 minutes (LSTM)



(b) Estimated demand 60 minutes (LSTM)



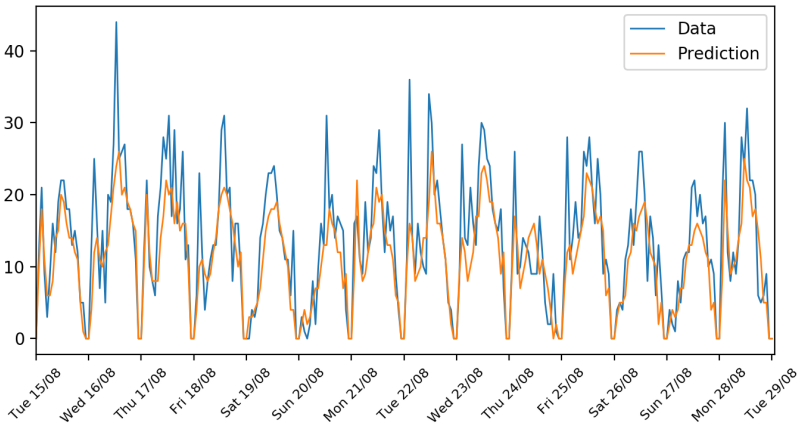
(c) Censored demand 60 minute (GRU)



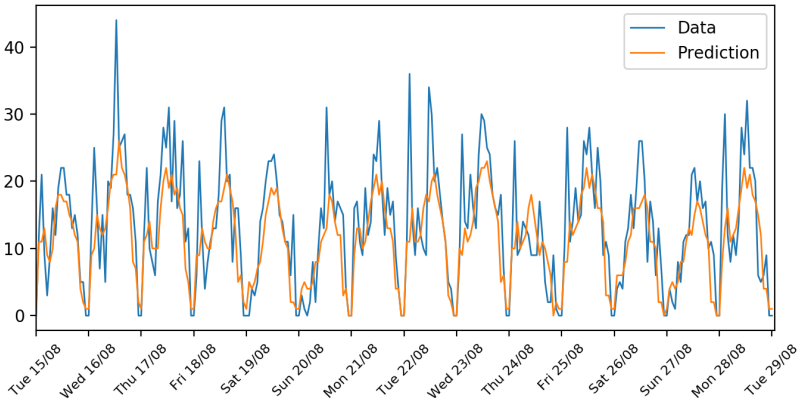
(d) Estimated demand 60 minutes (GRU)

The slope of the validation and training loss graphs indicate that the models are properly fitted.

Figure 6.8: Sample of training and validation loss for GRU



(a) LSTM Station 191 60 minutes Complete

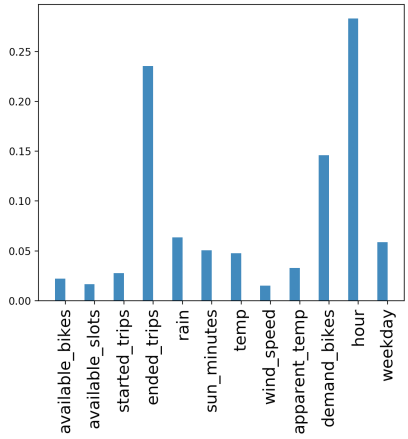


(b) GRU Station 191 60 minutes Complete

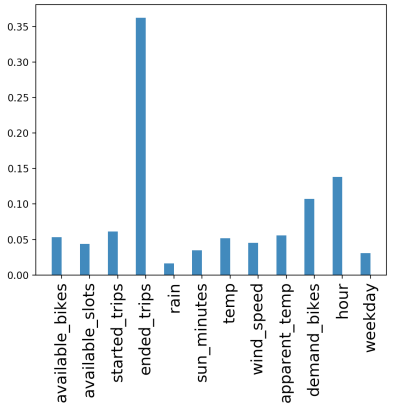
The RNN LSTM (a) is generally better at predicting morning peaks than the RNN GRU (b). This can be seen on Tuesday 15, Thursday 17, Sunday 20, Monday 21. and Monday 28. Both fail to predict the morning peak on Friday 25 and Tuesday 22.

Figure 6.9: Comparison of prediction quality RNN LSTM and RNN GRU

The results show that the RF generally achieves better prediction accuracy than the GBT. A further improvement of the GBT might be possible given more parameter tuning, as the GBT has more tunable parameters than the RF, and might be able to be better adjusted to the data. Figure 6.10 and 6.11 show the feature rank for both GBT and RF. The figures shows how each model weighs its decisions based on the features.

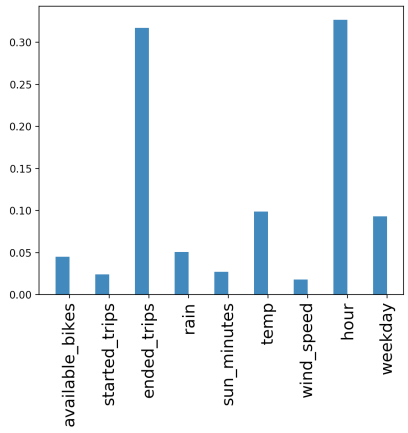


(a) GBT station 191 60 minutes complete

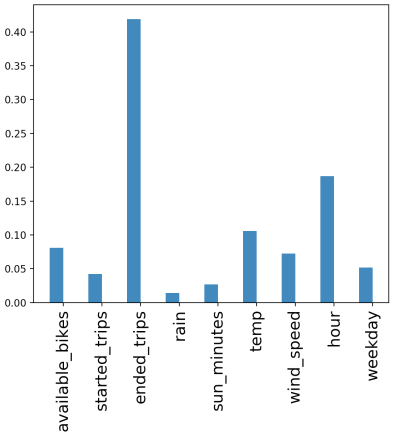


(b) RF station 191 60 minutes complete

Figure 6.10: Comparison of estimated demand feature rank RF and GBT



(a) GBT station 191 60 minutes complete



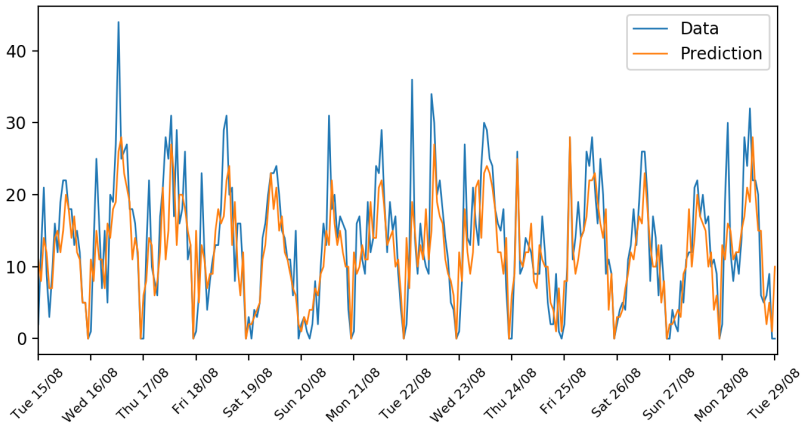
(b) RF station 191 60 minutes complete

Figure 6.11: Comparison of censored demand feature rank RF and GBT

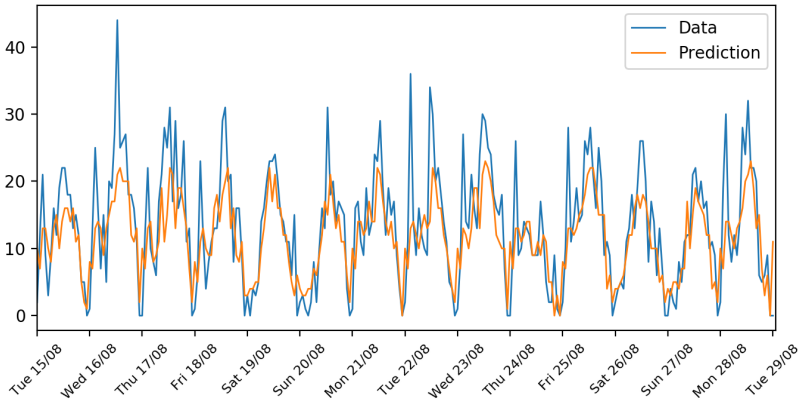
The ranking shows a clear difference in how the two utilize the features. The GBT mostly bases its decisions on the number of ended trips and the hour of the day, both for censored and estimated demand. This corresponds with earlier findings as the hour of the day was found to have a strong correlation with demand. The fact that the number of ended trips is weighted almost equally with the hour is an indication that this station often is empty, and that bikes that are parked here soon after are rented again. This theory is strengthened by the fact that station 191 is neutral, and most bike movements thus are made by users.

The RF show similar dependencies to the GBT but relies more on ended trips and less on the hour of the day. This, in addition to the RF achieving the best accuracy of the two, indicates that the number of ended trips is a better indication of future demand for this station.

Figure 6.12 show that the ensemble models struggle on the same aspects of the prediction as the RNNs. Both the RF and the GBT often underestimate the demand volume, while predicting the general demand curve relatively well. The GBT struggles more than the RF and often predicts lower than actual peaks, and higher than actual bottoms. The RF is also better at predicting the morning peaks than the GBT.



(a) RF station 191 60 minutes complete



(b) GBT station 191 60 minutes complete

The difference between the RF (a) and GBT (b) pronounced. The GBT is both worse at predicting morning peaks and predicting the correct demand volume. The GBT predicts less demand than the RF on Wednesday 16. Monday 28 and Saturday 26. The GBT fails to predict the morning peak on Friday 25. and Monday 28.

Figure 6.12: Comparison of prediction quality RF and GBT

Chapter 7

Conclusion

7.1 Conclusion

The goal of this thesis is to evaluate the performance of common machine learning methods for predicting station-level bike-sharing system (BSS) demand in the Oslo BSS. This was achieved by comparing the performance metrics of several time-series prediction models through three experiments.

Research Question 1

Which commonly used machine learning methods are the best candidates for predicting station-level demand in the Oslo BSS?

Of the most common machine learning methods for this and similar domains, RNNs and ensemble learning methods were found to be the best candidates for predicting station level demand in the Oslo BSS. Related work presented good results with Recurrent Neural Networks using LSTM and GRU blocks on station-level demand with data of similar type and quality to what is available in the Oslo BSS. Gradient Boosting Tree and Random Forest were found to provide the best results on system-level demand using similar features.

Research Question 2

How accurate are these methods in predicting demand in the Oslo BSS?

All models achieved good prediction accuracy above baseline predictions on the censored demand dataset. The LSTM RNN model was found to generally outperform the other models, considering only the results from E1 and E2. The best prediction accuracy is achieved with a 30-minute prediction-span, on the complete and limited dataset.

The highest accuracy is achieved by the LSTM RNN model on the 30-minute timeframe and complete dataset, with a mean accuracy of 0.551 RMSLE and 2.557 RMSE. The worst accuracy across all models is produced with the estimated demand dataset, where all models fail to learn any complex patterns due to noisiness.

Research Question 3

How does the performance of these methods differ?

There is a moderate difference in performance between the models. One of the most important differences observed is that the RNNs performed best with both the complete and limited dataset. The performance shows that they are able to generalize the patterns for both seasons and utilize all available data to improve accuracy, while the accuracy of the ensemble models worsened with more data. This also shows that the RNNs can learn more from the limited dataset than the ensemble models could.

The difference in prediction quality is less evident, and all models make good predictions and are able to foresee most spikes in demand. The RNN LSTM and RF models are better at predicting the sharp morning spikes and estimate the size of both these and the other spikes. The GBT and GRU models struggled with correctly predicting zero demand, and often predicted slightly above zero.

Testing with different prediction spans revealed a change in prediction accuracy but no change in the differences in performance among the models. All models performed better with the 30-minute prediction-span. The RNN LSTM and RF better predict the morning spikes with the 30-minute prediction-span, while all models struggled slightly more on the afternoon spikes.

None of the models achieved good accuracy on the estimated demand dataset.

7.2 Future work

7.2.1 Combined model

The architectures employed in this thesis uses a separate model for each station in the BSS in order to simplify parameter tuning and to keep the complexity low. This architecture can be cumbersome in a production environment, as a model has to be trained for each of the stations individually. Chen et al. (2017) provided a solution where one model is used to predict the rental and return demand for all stations in a BSS. This solution would likely work well with the Oslo BSS data and is a logical next step to employ the predictions in the BSS.

7.2.2 Effect of neighbor stations

The Oslo BSS is in constant change, and new stations are being introduced continuously throughout the season. The constant change creates challenges for the models as new stations changes the demand profile of existing nearby stations, making earlier data less representative of the new profile. Incorporating this proved difficult as no good solution was found to account for this change. However, related work has shown that the condition of nearby stations is an important feature, and is something that would likely improve predictions. This could be accomplished with a combined model (Rudloff and Lackner, 2014).

7.2.3 Separated demand by user group

The Oslo BSS provides day passes and season passes for rentals, which can have a potentially very different usage pattern. Related research has shown that separating these usage-groups can have a significant impact on prediction accuracy (Zhang et al., 2016), and it is likely this applies for the Oslo BSS as well.

Bibliography

Allende, H., Valle, C., 10 2017. Ensemble methods for time series forecasting 349, 217–232.

Breiman, L., 2001. Random forests. Machine Learning 45 (1), 5–32.

Breiman, L., 2017. Classification and regression trees. Routledge.

Chen, P. C., Hsieh, H. Y., Sigalingging, X. K., Chen, Y. R., Leu, J. S., 2017. Prediction of station level demand in a bike sharing system using recurrent neural networks. Vehicular Technology Conference (VTC Spring), 1–5.

Cho, K., van Merriënboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., Oct. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, pp. 1724–1734.

Dietterich, T. G., 2002. Ensemble learning. In The Handbook of Brain Theory and Neural Networks, 2nd ed. Cambridge, MA: The MIT Press.

Fishman, E., 2016. Bikeshare: A review of recent literature. Transport Reviews 36, 92–113.

Fishman, E., Washington, S., Haworth, N., 2013. Bikeshare: A synthesis of the literature. Transport Reviews 33, 148–165.

Frade, I., Ribeiro, A., 2013. Bicycle sharing systems demand.

Friedman, J., 1999. Stochastic gradient boosting. Technical report, Stanford University, Statistics Department.

Froehlich, J., Neumann, J., Oliver, N., 2009. Sensing and predicting the pulse of the city through shared bicycling. In: Proceedings of the 21st International Jont Conference on Artifical Intelligence. IJCAI'09. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1420–1426.
URL http://dl.acm.org/citation.cfm?id=1661445.1661673

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 249–256.

Gurney, K., 1997. An Introduction to Neural Networks. CRC Press.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation 9 (8), 1735–1780.

Jia, Z., Xie Gang, Gao, J., Yu, S., 2017. Bike-sharing system: A big-data perspective.

Kaltenbrunner, A., Meza, R., Grivolla, J., Codina, J., Banchs, R., 2010. Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. Pervasive and Mobile Computing 6, 455–466.

Liu, J., Li, Q., Qu, M., Chen, W., Yang, J., Xiong, H., Zhong, H., Fu, Y., 2015. Station site optimization in bike sharing systems.

Mickey, R. M., Greenland, S., 1989. The impact of confounder selection criteria on effect estimation. American Journal of Epidemiology 129 (1), 125–137.

Norvig, P., Russell, S. J., 2009. Artificial Intelligence: A Modern Approach. Pearson; 3 edition.

Ortiz-Rodriguez, J. M., del Rosario Martinez-Blanco, M., Viramontes, J. M. C., Vega-Carrillo, H. R., 2013. Robust design of artificial neural networks methodology in neutron spectrometry. In: Artificial Neural Networks-Architectures and Applications. InTech.

Racine, J., 2000. Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. Journal of Econometrics 99 (1), 39 – 61.
URL http://www.sciencedirect.com/science/article/pii/S0304407600000300

Rudloff, C., Lackner, B., 2014. Modeling demand for bikesharing systems: Neighboring stations as source for demand and reason for structural breaks. Transportation Research Record 2(2430), 1–11.

Shaheen, S. A., Guzman, S., Zhang, H., 2010. Bikesharing in europe, the americas, and asia. Transportation Research Record: Journal of the Transportation Research Board 2143 (1), 159–167.

Singhvi, D., Singhvi, S., Frazier, P. I., Henderson, S. G., O'Mahony, E., Shmoys, D. B., Woodard, D. B., 2015. Predicting bike usage for new york city's bike sharing system.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15 (1), 1929–1958.

Tran, T. D., Ovtracht, N., d'Arcier, B. F., 2015. Modeling bike sharing system using built environment factors.

Vogel, P., Mattfeld, D., 2011. Strategic and operational planning of bike-sharing systems by data mining–a case study. Computational Logistics, 127–141.

Wang, D., Cao, W., Li, J., Ye, J., 2017. Deepsd: Supply-demand prediction for online car-hailing services using deep neural networks.

Weigend, A. S., 1994. Time series prediction: forecasting the future and understanding the past. Santa Fe Institute Studies in the Sciences of Complexity.

Xu, J.-X., Lim, J., 2007. A new evolutionary neural network for forecasting net flow of a car sharing system. In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on. IEEE, pp. 1670–1676.

Yin, Y.-C., Lee, C.-S., Wong, Y.-P., 2012. Demand prediction of bicycle sharing systems. URL http://cs229.stanford.edu/proj2014/YuchunYin,ChiShuenLee,Yu-PoWong,DemandPredictionofBicycleSharingSystems.pdf

Yu, L., Wang, S., Lai, K. K., Feb. 2006. An integrated data preparation scheme for neural network data analysis. IEEE Trans. on Knowl. and Data Eng. 18 (2), 217–230. URL http://dx.doi.org/10.1109/TKDE.2006.22

Zhang, J., Pan, X., Li, M., Yu, P. S., 2016. Bicycle-sharing system analysis and trip prediction.

Appendix

		RNN LSTM											
		RMSE				RMSLE							
		60 Min		30 Min		60 Min				30 Min			
Station id	Trips count	Complete	Limited	Complete	Limited	Complete	Limited			Complete	Limited		
ARRIVAL													
279	24488	2.660838282	3.067350563	1.76490598	1.969533255	0.525173943	0.562450104			0.515869762	0.539438749		
234	37639	3.47	3.464558414	2.191612421	2.18940708	0.575673489	0.607897076			0.542749438	0.539579742		
204	55133	4.01	4.579686632	2.608947576	2.836197953	0.551474639	0.567717523			0.557232303	0.562263674		
277	81098	4.28351966	4.149705187	2.772543332	2.751264978	0.53019730	0.514176826			0.525085450	0.522930896		
253	99541	5.56065881	6.062884666	3.546727604	3.607957745	0.553495716	0.595908423			0.534291955	0.535657668		
DEPARTURE													
291	30101	2.956675938	3.150455648	1.91315143	2.04422718	0.666337040	0.675343409			0.603348181	0.620061117		
183	32159	3.377055224	3.393549277	2.11665567	2.116450608	0.651159288	0.649437574			0.598426636	0.591469850		
161	39775	3.48	3.746047563	2.342093004	2.452647552	0.658668626	0.693352969			0.623764174	0.644257429		
233	131173	6.659506709	6.682199825	4.230802055	4.207567823	0.570959621	0.570968122			0.555435134	0.560690256		
267	116229	6.14758691	6.256633791	3.927415825	3.822486976	0.57430692	0.590811327			0.536489192	0.557217071		
NEUTRAL													
190	55466	3.602123172	3.557699679	2.250521861	2.265767645	0.547491922	0.526437891			0.538859002	0.538713941		
245	30073	2.696214261	2.822703492	1.628967226	1.716231968	0.585229602	0.589937384			0.539320727	0.547686294		
210	33885	4.117350579	3.832124347	2.567048752	2.432151256	0.548633661	0.540371378			0.531385255	0.541804255		
251	70874	2.627595178	2.509220304	1.704959507	1.642568003	0.562905019	0.555045519			0.534019832	0.528393150		
191	80769	4.240790627	4.367575645	2.794196205	2.905816829	0.526089348	0.498008816			0.527025017	0.528594838		
	AVG Arrival	3.998	4.265	2.577	2.671	0.547	0.570			0.535	0.540		
	AVG Departure	4.119	4.243	2.651	2.705	0.637	0.647			0.595	0.604		
	AVG Neutral	3.472	3.404	2.149	2.138	0.560	0.552			0.537	0.543		
	AVG	3.993	4.109	2.557	2.597	0.575	0.583			0.551	0.557		

Table 7.1: Part 1 of censored demand results

RNN GRU							
RMSE				RMSLE			
60 Min		30 Min		60 Min		30 Min	
Complete	Limited	Complete	Limited	Complete	Limited	Complete	Limited
3.13811	3.079723279	1.803979548	2.018132657	0.5953524227	0.5718403225	0.5206926875	0.5474661247
3.57845	3.498068595	2.240393039	2.245549313	0.596243657	0.6247904007	0.5547055248	0.5564544117
4.17303	4.584190365	2.631437237	2.868954501	0.5637617705	0.5832829914	0.5675312441	0.566351564
4.500527901	4.276665331	2.860208128	2.844969838	0.548520349	0.5315381614	0.5376271734	0.5444084438
5.904663434	6.158884676	3.712273332	3.718019872	0.5753793725	0.6172273221	0.5642099765	0.5571798768
2.97444	3.14257	1.953057914	2.050746268	0.6690820407	0.6780855421	0.6104001321	0.6247487821
3.41757	3.42153	2.116344617	2.146071171	0.6637788186	0.6610579634	0.5978983948	0.5973394678
3.54701	3.77058	2.379213418	2.487013304	0.67021472	0.6857068044	0.6338197512	0.6547984537
6.805974046	6.798973877	4.326980025	4.273013937	0.6046049432	0.6159378418	0.5689511469	0.5868303429
6.229458377	6.267028199	4.084216758	3.877376296	0.5711425824	0.6082362544	0.5605519139	0.5679597906
3.65899	3.590052369	2.29823264	2.289851329	0.5697324561	0.5327490465	0.5528194625	0.5482757094
2.78705	2.936074812	1.697824488	1.778917908	0.597543087	0.604271513	0.5455780518	0.5613810523
4.20093	3.839200401	2.624716199	2.43044255	0.5772427394	0.5487171326	0.5538327344	0.5540787705
2.65256	2.517162911	1.72809167	1.628907493	0.5788961309	0.5589318649	0.542020978	0.5265855124
4.514882725	4.388363756	2.876560394	2.899246129	0.580536809	0.4968966244	0.5495552289	0.5290236489
4.259	4.320	2.650	2.739	0.576	0.586	0.549	0.554
4.186	4.283	2.694	2.739	0.652	0.660	0.603	0.616
3.549	3.455	2.207	2.166	0.582	0.562	0.551	0.555
4.139	4.151	2.622	2.637	0.597	0.595	0.564	0.568

Table 7.2: Part 2 of censored demand results

Random Forest												
RMSE						RMSLE						
60 Min		30 Min				60 Min				30 Min		
Complete	Limited	Complete	Limited			Complete	Limited			Complete	Limited	
3.03412	3.11266699	1.89941069	1.97943480	9		0.60550534	0.58996317	28		0.58180406	0.57889681	42
3.69735	3.47697417	2.30275534	2.26965422	28		0.63204365	0.60971052			0.60430547	0.59932723	44
4.44878	4.61845112	2.74177397	3.00218368	3		0.60782990	0.59340485	15		0.61348111	0.63439032	1
4.41692168	4.40562018	2.94664775	2.89157989	5		0.53768973	0.51812854	18		0.56388321	0.57616846	27
5.98419245	5.82664616	3.80424333	3.76176695	8		0.56542261	0.53685087	55		0.59600824	0.58273834	25
3.02734	3.17217	1.96005041	2.01901202	9		0.72221210	0.70911454	46		0.66859581	0.65345874	51
3.42558	3.24118	2.13916413	2.08770881	8		0.70885994	0.66457690	5		0.65579714	0.62795056	99
3.59311	3.82086	2.39840704	2.46967694	4		0.70081900	0.71974441	65		0.69181089	0.69573981	9
6.773515963	7.050046878	4.39385273	4.46500303	3		0.58570147	0.56335049	56		0.59485399	0.63274047	34
6.32532105	6.77741551	4.11244548	3.91475415	4		0.56539875	0.62222087	84		0.57809717	0.60022864	17
3.68955	3.31767185	2.35985211	2.25959	53		0.58134678	0.51579035	69		0.59194393	0.55833308	15
2.80614	2.87695112	1.70008759	1.82764890	6		0.63559495	0.62373148	14		0.59380114	0.60517319	02
4.15251	3.63930370	2.66341194	2.39097122	4		0.56863315	0.52420863	23		0.57018675	0.56054410	14
2.64883	2.20287476	1.74918373	1.53305744	1		0.59132178	0.51682801	72		0.56958940	0.51081420	63
4.427744227	4.72029548	2.94885146	2.92184706			0.55093567	0.53579630	18		0.57510891	0.59027903	12
4.316	4.288	2.739	2.781		0.588	0.570		0.592		0.594		
4.205	4.321	2.723	2.760		0.679	0.664		0.653		0.652		
3.549	3.278	2.241	2.159		0.595	0.555		0.585		0.575		
4.163	4.151	2.675	2.653		0.610	0.590		0.603		0.600		

Table 7.3: Part 3 of censored demand results

Gradient Boosted Regressor												
RMSE						RMSLE						
60 Min		30 Min				60 Min				30 Min		
Complete	Limited	Complete	Limited			Complete	Limited			Complete	Limited	
3.02791	3.182408538	1.82041383	1.909158381			0.587932140	0.588942302			0.547753189	0.552529608	
3.58869	3.499606525	2.21250402	2.204168990			0.635292985	0.633991330			0.578033818	0.581435873	
4.28133	4.569672090	2.65094027	2.894338490			0.624194656	0.607267484			0.608385574	0.608956689	
4.86020199	4.465783983	2.92650257	2.799785591			0.604726464	0.571458481			0.576047733	0.564272013	
6.34406760	5.978053582	3.71593132	3.714716020			0.637379750	0.611330461			0.596600962	0.584845710	
3.00612	3.06291	1.90020848	1.931911196			0.701430103	0.691527573			0.624412695	0.614466216	
3.30316	3.14988	2.06156716	1.990295259			0.692538158	0.651599780			0.622002036	0.588873026	
3.76462	3.67928	2.34473431	2.342620936			0.713250049	0.707482450			0.660172668	0.652128586	
7.12874149	6.909053512	4.32329950	4.262722665			0.648611612	0.618992541			0.603020581	0.607625398	
7.23429473	6.654136677	4.13803852	3.797215203			0.667415669	0.659948059			0.599921695	0.595365823	
3.40912	3.340358861	2.19411092	2.199288082			0.577516863	0.542906208			0.558551436	0.548855799	
2.81144	2.87648803	1.70696559	1.778953549			0.637331008	0.631653732			0.576453660	0.580275362	
3.86846	3.607962729	2.45069053	2.304374251			0.589032303	0.555675434			0.538900840	0.542973948	
2.36950	2.172646845	1.57457216	1.467611573			0.556648994	0.521569186			0.514512692	0.491595949	
4.76323856	4.584537281	2.82199362	2.811256453			0.622900088	0.560590985			0.575739895	0.556234020	
4.420	4.339	2.665	2.704	0.618	0.603	0.581	0.578					
4.301	4.200	2.657	2.632	0.689	0.667	0.627	0.616					
3.363	3.275	2.117	2.094	0.601	0.577	0.558	0.557					
4.251	4.116	2.589	2.561	0.633	0.610	0.585	0.578					

Table 7.4: Part 4 of censored demand results

Baseline												
RMSE						RMSLE						
60 Min			30 Min			60 Min			30 Min			
Complete	Limited		Complete	Limited		Complete	Limited		Complete	Limited		
4.36390	4.962711423		2.389070312	2.635083609		0.745520904	0.774046187		0.688350927	0.714589790		
4.75084	4.713659698		2.809205	2.858986851		0.767194394	0.779248483		0.694285315	0.699332939		
5.78711	6.434993249		3.397348241	3.742802022		0.763091268	0.782563937		0.727308888	0.735291140		
6.483222477	6.589368857		3.800116331	3.803571994		0.727048824	0.728725237		0.689564229	0.695774045		
8.492691032	8.692732788		4.672271224	4.833521324		0.766877159	0.780573407		0.702946476	0.696850134		
4.02163	4.38985		2.507779044	2.702398842		0.863713222	0.896612779		0.770724978	0.803595019		
4.31222	4.41926		2.645433846	2.730067953		0.819080448	0.823348207		0.738638649	0.747163088		
4.90905	5.26238		3.067485678	3.295954757		0.862730291	0.902160154		0.802893944	0.832917707		
9.044923698	9.193080042		5.494454329	5.536272349		0.792305043	0.796286875		0.702785197	0.720529871		
8.606989104	8.486336829		4.959502663	4.829948326		0.746577628	0.767159696		0.663545668	0.680393381		
4.66911	4.755878474		3.013586972	3.136042277		0.722833746	0.712324708		0.717239986	0.726435319		
3.80158	4.362768468		2.220278454	2.437297799		0.785442391	0.819571442		0.713371826	0.737231507		
5.43623	5.132638421		3.402070329	3.266900303		0.748299017	0.760443366		0.689608296	0.710752934		
3.50374	3.344413054		2.303305827	2.267541521		0.759068627	0.755014971		0.710038986	0.718051233		
6.083779143	6.416206441		3.710208181	3.835161422		0.736826667	0.749047346		0.693743142	0.698778393		
5.976	6.279		3.414	3.575		0.754	0.769		0.700	0.708		
5.572	5.816		3.429	3.566		0.834	0.855		0.754	0.776		
4.636	4.750		2.879	2.947		0.752	0.764		0.707	0.725		
5.618	5.810		3.359	3.461		0.774	0.788		0.714	0.728		

Table 7.5: Part 5 of censored demand results

		RNN LSTM											
		RMSE						RMSLE					
		60 Min			30 Min			60 Min			30 Min		
Station id	Trips count	Complete	Limted		Complete	Limted		Complete	Limted		Complete	Limted	
ARRIVAL													
279	24488	8.140410953	7.443496406		7.319035279	7.46498566		0.7825184032	0.76542743		0.6921428923	0.787839979	
234	37639	6.64	6.908240579		4.820141441	6.725581246		0.7774309704	0.7403336655		0.6162887402	0.765574807	7
204	55133	5.31	4.894144156		4.18500776	4.742775191		0.5938839063	0.6185266073		0.4798881029	0.597188164	2
277	81098	5.686420973	5.593494771		4.253961469	5.423555748		0.5676882851	0.5633049742		0.4192485791	0.547748456	9
253	99541	7.491485358	7.428582598		5.373891908	6.601522363		0.6442780713	0.6446795598		0.4658532247	0.594900155	8
DEPARTURE													
291	30101	7.79558177	8.19674128		5.834698065	7.616456203		1.035715007	1.058791139		0.8600182514	1.036036229	
183	32159	8.86	9.348481115		7.828182132	9.190191174		0.976411082	0.9595294921		0.8660013157	1.009222342	
161	39775	9.546398237	9.723007998		7.343524904	9.138890159		1.030343531	1.060971983		0.8705408053	1.045712495	
233	131173	10.31205248	10.53819372		8.056380205	9.508865815		0.5671270555	0.5962677301		0.4777183673	0.570803899	8
267	116229	9.696804216	9.766994236		7.602964839	8.904051453		0.6259843235	0.6521512148		0.5082300359	0.618929501	7
NEUTRAL													
190	55466	6.506302042	6.36536943		5.009562421	5.993070387		0.6180140959	0.65674272		0.4988374289	0.626535080	5
245	30073	6.67638883	6.722627324		6.345471327	6.704682104		0.8263896502	0.8248421342		0.7257874171	0.822863661	2
210	33885	6.017597014	5.968860206		4.607766945	5.488046058		0.6077685792	0.6213979842		0.485781592	0.602555595	3
251	70874	6.386615751	6.366851906		4.854861745	5.886925273		0.8180462802	0.8328927104		0.6599235914	0.822364362	8
191	80769	6.625265468	6.529970372		4.941248183	5.924463906		0.5481179368	0.5482332728		0.4175749686	0.516687948	
	AVG Arrival	6.653	6.454		5.190	6.192		0.673	0.666		0.535	0.659	
	AVG Departure	9.242	9.515		7.333	8.872		0.847	0.866		0.717	0.856	
	AVG Neutral	6.442	6.391		5.152	5.999		0.684	0.697		0.558	0.678	
AVG		7.590	7.608		6.045	7.193		0.743	0.751		0.613	0.740	

Table 7.6: Part 1 of demand results

RNN GRU												
RMSE					RMSLE							
60 Min		30 Min			60 Min				30 Min			
Complete	Limted	Complete	Limted		Complete	Limted			Complete	Limted		
8.20096	7.492733355	7.623123201	7.364657001		0.8500137798	0.796117886	7		0.747848669	9 0.770382680	9	
6.90731	7.034304438	5.562389156	6.388788588		0.8589264726	0.771357339	3		0.720962827	6 0.721347752	8	
5.77545	5.259404014	4.382727018	4.389524043		0.6711304816	0.669293998	4		0.515645895	7 0.552858970	8	
6.294771086	6.350206369	4.61326427	4.775400426		0.6105344236	0.64109563			0.469470350	1 0.507214511	5	
8.341598049	8.169703903	5.555554719	5.731175943		0.7064438063	0.738734746			0.500318619	3 0.528243964	8	
7.96070	8.21428	6.004445118	6.875048585		1.054790257	1.088787658			0.874033367	3 0.963520398		
8.95829	9.44756	8.320351128	9.016125069		1.059474038	1.010626266			0.933822424	7 0.968919422		
9.59755	9.87288	7.529208713	8.449236897		1.066425331	1.093628666			0.92125217	1.00546578		
11.16293026	11.58957589	8.094935732	8.606596831		0.6576920762	0.685158739	8		0.500246466	5 0.536208590	4	
10.56744109	10.60959294	7.553961239	8.172250276		0.7200707256	0.734930175	4		0.515776201	4 0.561793481	2	
6.83045	6.742037337	5.140033101	5.3574253		0.6717996201	0.697802982	8		0.524335037	4 0.564411413	9	
6.93656	6.884771516	6.52905949	6.560166223		0.8740129734	0.858987810	8		0.754186413	2 0.766968682	1	
6.38180	6.383930376	4.573590147	4.883336203		0.6732835189	0.69809999	8		0.481376663	8 0.532303879		
6.5356333	6.563402902	4.84449674	5.247567835		0.8492276451	0.874411574	7		0.662552951	1 0.733371986		
7.042618531	7.063032134	5.027290114	5.394248766		0.6102062947	0.620908989	4		0.441075227	4 0.475712276	2	
7.104	6.861	5.547	5.730		0.739	0.723			0.591	0.616		
9.649	9.947	7.501	8.224		0.912	0.923			0.749	0.807		
6.745	6.727	5.223	5.489		0.736	0.750			0.573	0.615		
7.994	8.004	6.268	6.659		0.806	0.807			0.651	0.691		

Table 7.7: Part 2 of demand results

Random Forest												
RMSE						RMSLE						
60 Min		30 Min				60 Min				30 Min		
Complete	Limted	Complete	Limted			Complete	Limted			Complete	Limted	
7.70790	8.069486647	6.536219557	6.939475515			0.798765120 6	0.801021636 9			0.655660964 5	0.669717274 5	
5.95409	6.663666274	4.908597201	5.121899825			0.685400763 4	0.767045099 3			0.561003546 7	0.637562334 9	
5.33509	5.164726021	4.461019974	4.503274455			0.564747247 3	0.569870504 2			0.476529372 9	0.496501306 8	
5.178716199	4.912307665	4.636566158	4.538401562			0.477342272 8	0.488356267 6			0.409776061	0.431560173 7	
6.524130234	6.401489948	5.778478277	5.662812646			0.506850904 2	0.513496533 9			0.441270731 2	0.459059057 4	
7.59013	7.72907	6.546057211	6.690102192			1.123694529	1.159054905			0.986406768 5	1.012690215	
9.30986	9.48185	7.923172676	7.730062838			1.025214608	1.031537892			0.842322897 6	0.840833444 9	
9.46121	9.57933	7.987732546	8.038858181			1.128437657	1.17372599			0.984180238 1	1.023578897	
9.886422631	10.10353309	8.930585053	8.875909679			0.555675064 5	0.585865295 9			0.503769860 2	0.520763250 9	
9.383203934	9.451910556	8.539009086	8.643446566			0.541473659 5	0.585056565 8			0.481653466	0.505501145 7	
6.26466	6.300176002	5.557785131	5.487820293			0.623531645 5	0.648643800 8			0.548991866 8	0.567884367 5	
7.33635	6.948036555	6.618299274	6.123524402			0.851730350 1	0.841615551 7			0.746809252 5	0.737477530 7	
5.91704	5.896484477	5.028550875	5.089734377			0.620128542 1	0.647895087 6			0.549510199 9	0.569574457 8	
6.343478682	6.379334224	5.487187293	5.449400844			0.892336482 8	0.913273904 8			0.790239396 8	0.795981039 4	
6.328276584	6.341569803	5.504708208	5.386098061			0.512051704	0.527827425			0.443081392 1	0.451234779 7	
6.140	6.242	5.264	5.353			0.607	0.628			0.509	0.539	
9.126	9.269	7.985	7.996			0.875	0.907			0.760	0.781	
6.438	6.373	5.639	5.507			0.700	0.716			0.616	0.624	
7.373	7.439	6.419	6.419			0.731	0.755			0.630	0.652	

Table 7.8: Part 3 of demand results

Gradient Boosted Regressor												
RMSE						RMSLE						
60 Min		30 Min				60 Min				30 Min		
Complete	Limted	Complete	Limted			Complete	Limted			Complete	Limted	
7.62656	7.880020634	5.723762754	5.831605447			0.8034110928	0.819509707	4		0.629424145	0.630017847	4
5.79939	6.452216032	4.485237547	4.834137484			0.7194280211	0.711178823	9		0.560346897	0.566968849	1
5.16474	4.974202882	4.160704231	3.947222289			0.5673998658	0.578509788	7		0.464855168	0.469339221	1
5.135400646	4.93238567	4.092754767	3.998784745			0.4973535831	0.505701604			0.406417088	0.425190272	6
6.639493104	6.434289925	5.570324167	5.392134054			0.5533171284	0.569613222			0.454445516	0.472233126	3
7.46319	7.87851	5.472951514	5.708467245			1.107752987	1.159118949			0.864524095	0.891357332	2
8.98348	10.29847	6.864132639	7.200287952			1.011142061	1.044467327			0.789889091	0.811345505	7
9.17308	9.62531	6.722383015	7.021241112			1.124890705	1.178591257			0.879433526	0.914118779	6
9.9870762	10.23470508	7.853171435	7.926328426			0.580097366	0.606475684	8		0.491557977	0.516320581	9
9.404342689	9.58760553	7.395506709	7.446661457			0.5874675104	0.623318133	4		0.477750253	0.505036955	9
6.21729	6.279830612	4.826313633	4.864372376			0.6340919395	0.669670418	8		0.509998778	0.535757623	2
6.98457	6.741355678	5.056506131	4.858244417			0.8594959849	0.855751045	2		0.681114358	0.683814358	
5.87806	5.797431588	4.496989839	4.425227513			0.6127551625	0.636626298			0.498289251	0.516577231	1
6.234873703	6.25592968	4.643436076	4.629201837			0.8815964702	0.912725882			0.683070901	0.701915012	2
6.21129322	6.05782212	4.837538488	4.725407266			0.5208367599	0.526736149	4		0.424769291	0.431600862	
6.073	6.135	4.807	4.801			0.628	0.637			0.503	0.513	
9.002	9.525	6.862	7.061			0.882	0.922			0.701	0.728	
6.305	6.226	4.772	4.700			0.702	0.720			0.559	0.574	
7.266	7.470	5.594	5.650			0.743	0.766			0.593	0.611	

Table 7.9: Part 4 of demand results

Baseline												
RMSE						RMSLE						
60 Min		30 Min				60 Min				30 Min		
Complete	Limted	Complete	Limted			Complete	Limted			Complete	Limted	
4.36390	4.99541073	8	2.38907031	2	2.64531783	0.7455209043	0.7777094118			0.6883509276	0.7213258734	
4.75084	4.70695800	1	2.809205		2.83072374	9 0.7671943947	0.8325592525			0.6942853151	0.7282631558	
5.78711	6.22492933		3.39734824		3.56432087	3 0.7630912683	0.7881604491			0.7273088881	0.7256242807	
6.483222477	6.60228854	8	3.80011633		3.81250156	7 0.7270488244	0.7508864279			0.6895642292	0.7177873691	
8.492691032	8.75187470	8	4.67227122		4.67945495	5 0.7668771594	0.7946028486			0.7029464767	0.7133161953	
4.02163	4.58563		2.50777904	4	2.68779853	4 0.8637132229	0.9469666472			0.7707249788	0.8088395201	
4.31222	3.99190		2.64543384	6	2.46302938	7 0.8190804486	0.8186402994			0.7386386492	0.7432504427	
4.90905	5.34131		3.06748567	8	3.28989277	9 0.8627302912	0.9523791071			0.8028939442	0.8499777678	
9.044923698	8.80233728	6	5.49445432	9	5.36515734	7 0.7923050435	0.820524562			0.7027851974	0.7312344021	
8.606989104	7.94556531	4	4.95950266	3	4.49614740	5 0.7465776288	0.7922031352			0.6635456687	0.6952494874	
4.66911	4.69736517	1	3.01358697	2	3.11424499	9 0.7228337467	0.7213731019			0.7172399862	0.7419509199	
3.80158	4.37774233	3	2.22027845	4	2.43587484	0.7854423919	0.828175692			0.7133718263	0.7485073466	
5.43623	4.89268108	9	3.40207032	9	3.15424895	1 0.7482990172	0.7584842165			0.6896082961	0.7295861184	
3.503738966	3.20494330	2	2.30330582	7	2.16942896	1 0.759068627	0.7651012069			0.7100389867	0.7217416327	
6.083779143	6.64790939	2	3.71020818	1	3.79743193	4 0.7368266676	0.7745036607			0.693743142	0.7014094606	
5.976	6.256		3.414		3.506	0.754	0.789			0.700	0.721	
6.179	6.133		3.735		3.660	0.817	0.866			0.736	0.766	
4.699	4.764		2.930		2.934	0.750	0.770			0.705	0.729	
5.745	5.840		3.414		3.426	0.778	0.814			0.715	0.743	

Table 7.10: Part 5 of demand results