Linköping University | Department of Computer and Information Science
Master thesis, 30 ECTS | Datateknik
2018 | LIU-IDA/LITH-EX-A--18/003--SE

Predicting Personal Taxi Destinations Using Artificial Neural Networks

Fredrik Schlyter

Supervisor : Héctor R. Déniz (LiU), Jonas Sköld (Bontouch)
Examiner : Jose M. Peña

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

Taxi Stockholm is a Swedish taxi company which would like to improve their mobile phone application with a destination prediction feature. This thesis has created an algorithm which predicts a destination to which a taxi customer would like to go. The problem is approached using the KDD process and data mining methods. A dataset consisting of previous taxi rides is cleaned, transformed, and then used to evaluate the performance of three machine learning models. More specifically a neural network model paired with K-Means clustering, a random forest model, and a k-nearest neighbour model. The results show that the models that were developed in this thesis could be used as a first step in a destination prediction system. The results also show that personal data increase the accuracy of the neural network model and that there exists a threshold for how much personal information is needed to increase the performance.

Acknowledgments

I would like to thank Bontouch and Taxi Stockholm for making it possible to do this thesis. A special thanks to my Bontouch supervisor Jonas Sköld. I also extend my sincere gratitude to my supervisor Héctor Déniz and my examiner Jose Peña from Linköping University. Lastly I would like to thank my family and my friends for the tremendous amount of support that I have received during my thesis project.

Fredrik Schlyter

Contents

List of Figures

List of Tables

1 Introduction

This chapter gives an overview of what this thesis has attempted to accomplish and a motivation for why this thesis has been conducted. The research questions are presented n one section and after that there is a section about the delimitations of the project.

1.1 Motivation

In the Taxi Stockholm mobile phone application all the destination entries have to be entered manually without any input from the application itself. This is a process which is not very user friendly and considering how much information is being stored about each user, it should be possible to improve this process. Uber which is a competitor to Taxi Stockholm has a destination prediction feature which gives the user travel suggestions depending on previous travel history and nearby popular destinations.

According to Uber more then 50 percent of all destination entries are done through the destination prediction feature[40]. Such a feature would save the user from the hassle of typing in the address manually, instead it would only require one tap to pick a destination. It would save time for the user and resources for the backend system by reducing the amount of requests containing spelling mistakes. The whole user experience would feel more personal if you receive suggestions of locations tailored from how you have traveled before instead of being greeted with an input prompt.

1.2 Aim

This thesis was requested by the Swedish taxi company called Taxi Stockholm. They provided the project with a dataset consisting of approximately 600,000 taxi rides collected around the city of Stockholm.

The aim of this thesis was to use data mining techniques in order to create and evaluate the underlying system behind the destination prediction feature. The input of this system consists of spatial, temporal, and metadata regarding the taxi ride and the output is the predicted destination. The goal was to first create a model based on all the taxi rides inside a dataset provided by Taxi Stockholm. Once that model was completed the final goal was to make it more personal by increasing the usage of personal information. The primary goal of the destination prediction feature is to increase customer satisfaction.

1.3 Research Questions

The goal of this thesis was to implement a neural network model which is capable of generating a destination prediction for a taxi ride. Once that goal was completed the next goal was to make the prediction more personal by increasing the usage of personal information. The following research questions were derived from the goals of this thesis.

1. Is it possible to predict a taxi ride destination using the model produced in this thesis and previous taxi ride history?

2. Does personal data improve the destination prediction of the model when compared to the predictions derived from general data?
3. Is there a threshold for when a model using personal data becomes more efficient than a model using general data?

1.4 Delimitations

The dataset used throughout this thesis will be the Taxi Stockholm dataset since they are the ones who requested this study. The methodology of the thesis will follow the knowledge discovery in databases process which is common practice within the scientific community. Neural networks will be used as the primary data mining method, however the neural network will be complemented by K-Means clustering to further improve the results.

1.5 Disposition

The following chapter introduces the reader to the theoretical background which is needed in order to understand what has been done throughout the process of this thesis. The third chapter explains the methodology which has been followed. The fourth chapter displays all the results gathered throughout this thesis. The fifth chapter contains a discussion about what has been done in this thesis and about what could be done in future work. The final chapter consists of the conclusions that were drawn at the end of this thesis project.

2 Theory

This chapter covers the theoretical background that was needed in order to conduct the study of this thesis. First an introduction to the knowledge discovery process is given, which has been the methodology of this thesis. The following sections presents different techniques for cleaning and manipulating the dataset. After the section about the data there is a section about machine learning and one section about neural networks which is the data mining method used in this thesis. Then there is a section which brings up evaluation techniques, one section which talks about recommender systems, and one section which looks at similar work.

2.1 Knowledge Discovery In Databases

Knowledge Discovery in Databases (KDD) is a process which is employed in order to extract knowledge from a set of data[15]. The amount of data that is made available due to technology has recently increased and there is often hidden knowledge in the data which requires manual data analysis to be found. The person analyzing the data often needs expert knowledge within the domain from which the data has been gathered. The large amount of data can be difficult for one person to digest and as such due to the increasing amount of available data, there is also an increasing need for better knowledge discovery techniques[10].

The methodology used during the course of this thesis is very similar to the KDD process and the steps that were followed are described below and are illustrated in figure 2.1.

Data selection or data addition is the process of removing superfluous data and adding new data to the dataset. This step includes finding which data is available and if there is any additional data that needs to be added[24].

Data cleaning or data preprocessing is the task of making sure that the dataset contains data which is reliable. Removal of faulty values such as certain features containing null values or limiting data to certain geographical locations[24].

Data transformation or data aggregation involves creating features and transforming data from the available dataset. It can also be thought of as generating comprehensible features from raw data. This step can be time consuming but the reward is an increased knowledge of the data and an increased performance[24].

Data mining is a step consisting of three tasks. The first task is to determine what type of data mining is to be used, such as classification, regression, or clustering. The chosen strategy should help accomplishing the goal of the process which could be predictive or descriptive data mining. The second task is to decide a method to use in order to achieve the goals described in the previous task. If precision is an important property of the model, then neural networks could be an appropriate method. The third task is to implement and tune the chosen model[24].

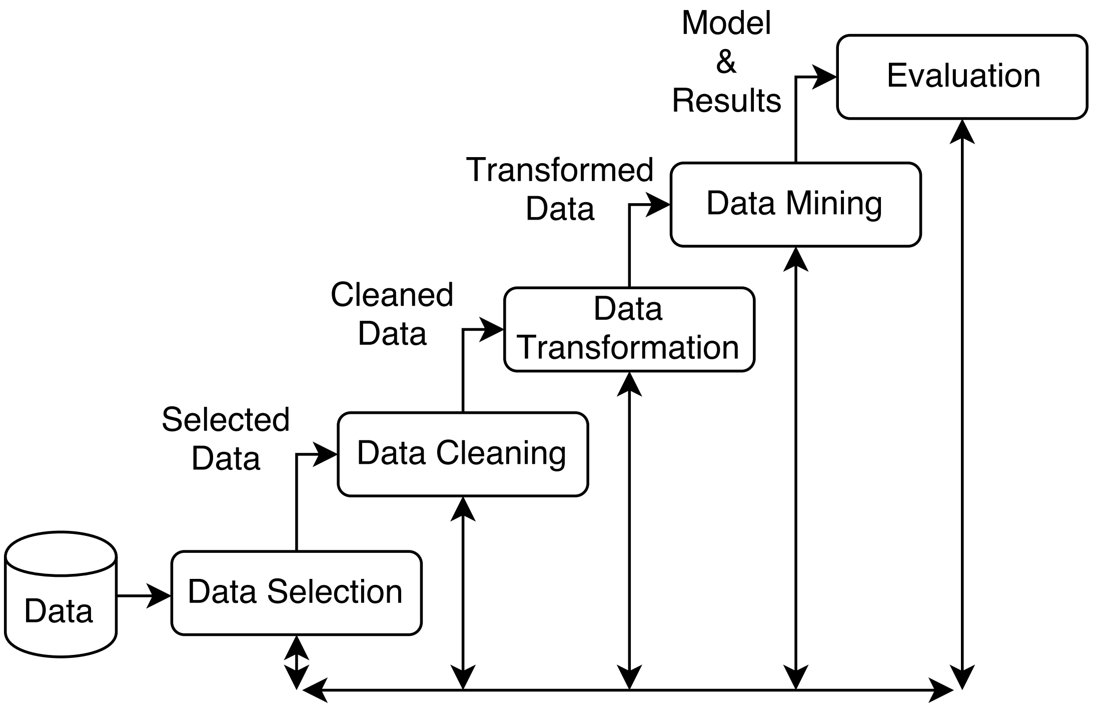


Figure 2.1: The knowledge discovery process (Source: [24])

Evaluation of the final model is an important step to make sure that the initial goals have been achieved. The usefulness of the model is also evaluated and the effects of the preprocessing steps on the result are observed[24].

Note that the first and final step of the KDD process has been left out. The first step consists of gathering domain knowledge about the problem that is to be solved. The final step is to incorporate the knowledge generated by the model into systems which benefit from the knowledge[24]. KDD is an iterative process which means that after each task it is possible to go back and change decisions that were made during previous tasks. The KDD process shares similarities with another process called CRoss Industry Standard Process for Data Mining (CRISP-DM) which is a simpler process compared to KDD[43]. It was decided to use a methodology similar to KDD since it seems to be the established model within the scientific community[35].

The first steps in the KDD process describes how to refine and improve the original data and as such in the next section the data which was used during the course of this project will be presented in greater detail.

2.2 Data Selection

One of the key components of traditional data mining methods is the dataset. The type of data that is available to you is instrumental when designing a satisfactory algorithm. The data played a key role in how this thesis was executed and three different datasets were investigated during the literature review process. The primary dataset of this study is a dataset from the Swedish taxi company Taxi Stockholm which consist of approximately 670,000 taxi rides. It will be used in order to train and evaluate the machine learning algorithm of this thesis. The original data set contains the information presented in table 2.1. The most important feature next to the departure and destination address is the user identification which makes it possible to train a personal model.

Feature	Description
user_id	A unique user Id which can be used to identify a user's travel history. The Id is anonymous.
job_id	A unique booking Id
from_string	The departure address of the taxi ride as a string. Contains the name of the street, the number and the municipality.
from_zone	The Taxi Stockholm departure zone.
to_string	The destination address of the taxi ride as a string. Contains the name of the street, the number and the municipality.
to_zone	The Taxi Stockholm destination zone.
date	The date of the taxi ride in big-endian format (YYYY-MM-DD).
time	The start time of the taxi ride in the extended ISO 8601 format (HH:MM:SS).
passenger	An encrypted string of the passenger name for confidentiality
phone	An encrypted string of the passenger phone number for confidentiality
origin	A string containing information about where the booking was made. For example if it was from an iPhone, an Android device or the website.
pf_booking_type	Contains information about whether the ride was ordered in advance or if it was ordered directly.

Table 2.1: Taxi Stockholm Dataset Content

Two features which makes the Taxi Stockholm dataset unique from the other datasets that have been found is the origin feature and the booking type feature. Especially the booking type feature could have a big impact on the performance of the algorithm. The Taxi Stockholm dataset is not open source and was given explicitly for this study.

There exist at least two other datasets which, in contrast to the Taxi Stockholm data, are open source. The first is a dataset from the city of Porto which consists of approximately 1.7 million taxi rides. The interesting aspect of the Porto dataset is that it contains a complete GPS trace of the taxi ride. It also has a user identification which would enable the possibility of personal recommendations. However the Taxi Stockholm dataset was chosen over the Porto dataset since the study was requested by Taxi Stockholm.

The second is a huge dataset from New York City with millions of taxi rides from each month of the year going back as far as 2009. This dataset would be good for a general taxi destination prediction algorithm. Unfortunately it does not contain any user identification, which prohibits it from being used when attempting to give a personal recommendation. Due to this the Taxi Stockholm dataset was chosen over the New York City dataset.

Feature Selection

Once a dataset has been chosen it is important to decide which parts of the dataset should be included and which parts should be neglected. If you are going to try and predict housing prices, then you might want to have house price and house size as two features. The algorithm would then be able to predict the price of a house depending on its size. A model which only relies on the price of the house and the size of the house in order to generate a price estimation, might not perform well in reality since there are several other factors which have an impact on the price of a house. If you add more features which are relevant to the problem that you try to solve, such as the number of bathrooms, then most likely the model will better represent how much the house actually costs.

However this does not necessarily mean that the more features you have, the better the prediction will be. There exists a problem called overfitting which is when the size of the

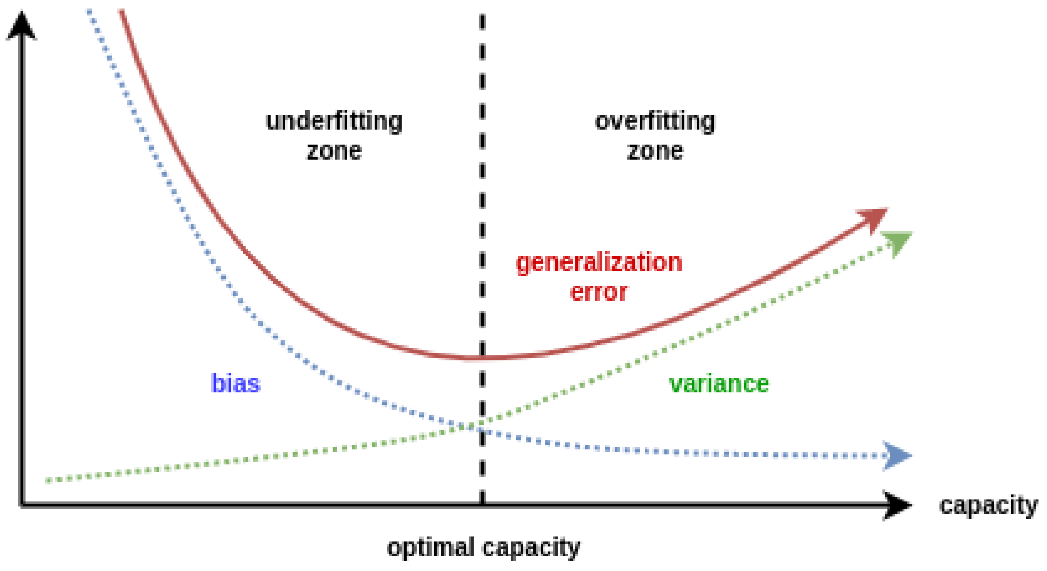


Figure 2.2: Relationship between model capacity, bias, variance, overfitting, and underfitting (Source: [34])

model is very big and the amount of training data is very small[16]. If this happens then the model will fit the training data perfectly but fail to generalize to new data. On the other hand there is another problem called underfitting which is when you have too few features in order to get a representative result from the algorithm. So there is a constant challenge when developing a model to have a good balance between the number of features, the size of the dataset, and the complexity of your model.

Another way of looking at overfitting and underfitting is through bias and variance. A model with high bias is equal to having a model that is underfit while a model with high variance is equal to having a model that is overfit. The optimal model has a low bias and a low variance which results in a low total error as seen in figure 2.2. High variance can be seen in complex models with insufficient training data while high bias is caused by having a model that is too simple[11].

Once the desired dataset has been chosen it is important to make sure that the features have a format which is useful. The following section will bring up the data transformation techniques used throughout this thesis.

2.3 Data Transformation

Once the desired features have been selected they can be transformed into values which better fit a specific model. The 1-of-K coding scheme or one hot encoding which it is sometimes called is a common way of representing data in classification problems. If you have K number of classes called C_k they can be represented as a binary vector of size K. All elements will be zero except for element k which is the class that the vector is supposed to represent[2]. 1-of-K encoding was used in this thesis to convert some of the features from discrete values into a binary vector representing different states.

Normalization of the data is a technique which is used for several different reasons. One of the reasons is to make sure that the result does not get impacted by the measurement unit of the data. Another important reason is that if there exist features within the dataset that has large values it is possible that features with small values will receive less weight[15]. In the scientific literature the words standardization and normalization sometimes have the same meaning and sometimes they do not have the same meaning. Henceforth equation (2.1) and equation (2.2) will represent the definition of the two different normalization techniques used throughout this thesis[15].

Equation (2.1) represents the zero-mean normalization for a feature vector $X = \{x_1, ..., x_n\}$ with the mean $\bar{x}$ and standard deviation σ. This method of normalizing the data can have a positive effect on the result if there are plenty of outliers in the data[15]. The coordinates were transformed using zero-mean normalization in this thesis.

$$x'_i = \frac{x_i - \bar{x}}{\sigma} \tag{2.1}$$

Equation (2.2) represents the min-max normalization for a feature vector $X = \{x_1, ..., x_n\}$. The current minimum and maximum values are min_X and max_X respectively, and the new minimum and maximum values are min_Y and max_Y respectively. One important property of the min-max normalization is that it keeps the original relationship between the values intact[15]. Originally the user identification feature had very large values and min-max normalization was used to convert it into a smaller number.

$$x'_i = \frac{x_i - min_X}{max_X - min_X}(max_Y - min_Y) + min_Y \tag{2.2}$$

Another important part of the data transformation process is to divide the dataset into three different datasets. One dataset is called the training set which is only used for training of the algorithm. One dataset is called the validation set which is used to evaluate the performance between different models when tuning the model hyperparameters. The third dataset is called the test set and the result of the test set can be used to measure how well the model generalize on new data.

There exist several different methodologies that recommend how to split the data in order to train the best possible model. One traditional method is to create a 33/33/33 percent split between training/validation/test datasets, another method is to create a 80/20 percent split between the training and the test datasets[8].

Another method which is taught by Andrew Ng in his machine learning course at Stanford is to split the data into a 60/20/20 percent split between training/validation/test datasets. The preliminary data processing conducted will be discussed in greater detail in the method chapter of this thesis.

2.4 Machine Learning

Machine learning is a field within Artificial Intelligence (AI) which recently has gained a lot of interest within the technology industry by proving what researchers previously thought impossible, to be possible[36].

> *"In particular, we define machine learning as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty"* - Kevin P. Murphy[26]

The quote by Kevin P. Murphy is a good definition of what machine learning actually is. Machine learning has proven helpful within areas such as image classification, email spam filtering, and face recognition[26]. Thanks to the development of deep learning, which is a subdivision within machine learning, there have been breakthroughs within natural language processing and health care[23].

One alternative to machine learning is the knowledge-based approach which implies that the existing knowledge is hard-coded into a rule set which the AI must follow[14]. The knowledge based approach fails to model uncertainty while machine learning, which is a probabilistic approach, succeeds at the same task. The knowledge based approach makes the

AI limited to the rule set and it is hard to cover all the potential events which might arise. Instead machine learning has gained a lot of traction because it allows the AI to create its own rule set from training data.

Machine learning is a broad term and can be divided into several different subdivisions. There is a lot of terms being thrown around such as supervised and unsupervised learning, and several different techniques such as linear regression, logistic regression and neural networks. If you are new to machine learning or to a certain area within machine learning it can be difficult to know the meaning of some of the expressions used in this thesis. Thus a short explanation of the different techniques and methods will be presented in the following subsections.

Supervised Learning

In supervised learning a machine learning model is fed with a labeled dataset, this means that an algorithm is given a training example and an expected output[2]. An euphemism for how it works is that the algorithm processes data much faster then any human and as such is capable of learning years of knowledge in a matter of minutes. If you let such an algorithm train for a couple of days then it is possible that it will have accumulated more experience then any human ever could in a lifetime.

There is two common applications of supervised machine learning called classification and regression models. Classification is when you teach an algorithm to predict the category or class of an object from a limited amount of classes. An example of an area which use classification algorithms is computer vision where the algorithm is supposed to guess what the image represents. Then there is regression which is when the algorithm tries to predict a continuous output variable such as housing prices[2].

Unsupervised Learning

In contrast to a supervised machine learning model, an unsupervised model is only fed with a training example and no expected output[2]. An unsupervised learning algorithm can be used to find similarities in the data, this technique is called clustering. The algorithm will use unlabeled data and try to divide the data into groups or categories depending on certain similarities between the samples[26]. If you have a dataset which consist of the height of different people, then the algorithm might label people below a height threshold into one group and the people above the threshold into one group, without any previous knowledge of what a short or a tall person is.

There exist many different types of clustering algorithms. One of the simplest types of clustering consists of methods called partitioning algorithms which includes a popular technique called K-Means. The K-Means clustering algorithm was used in this thesis in order to create clusters of all the taxi rides.

The K-Means clustering algorithm was first published in 1955 which is over 60 years ago, however it still remains relevant and is commonly used to tackle clustering problems[19]. The idea behind the algorithm is to minimize the sum of the squared error between the mean of a cluster and the points within that cluster. Then the primary objective is to minimize that function for all the clusters. A version of the K-Means algorithm can be seen in equation (2.3), the objective is to minimize the function in order to find the optimal set of clusters. One important task when using K-Means is the choice of initial clusters, if chosen wisely it can help the algorithm to avoid getting stuck in a local minimum.

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - \mu_k||^2 \tag{2.3}$$

A written description of how the K-Means algorithm work can be seen below for a dataset D and k clusters[19].

1. Select an initial set of k clusters from the data points in D.
2. Generate a new set of k clusters by assigning each data point in D to its closest cluster center.
3. Calculate k new cluster centers.
4. Repeat steps 2 and 3 until cluster memberships stabilizes.

Cost Function

One important component in the machine learning model is the cost function or loss function depending on the context. The purpose of the cost function is to give a quantitative measure of how well the model performs. There exist several different cost functions which perform better or worse depending on the problem formulation and the network architecture. One common loss function is called mean squared error which can be seen in equation (2.4).

$$MeanSquaredError = \frac{1}{n}\sum_{i=1}^{n}(Y_i' - Y_i)^2 \tag{2.4}$$

While the mean squared error is a good loss function when calculating the distance between two points, it would produce a small error when calculating the distance between two coordinates. The small error originates from the fact that the mean squared error does not take the curvature of the Earth into account when calculating the distance. The Haversine formula has been used for hundreds of years as a navigational tool and it calculates the great-circle distance between two points on the surface of a sphere[4]. It was also recommended by the creators of the 2015 Kaggle Taxi prediction challenge which solved a similar problem as this thesis[9]. The formula can be seen in equation (2.5). The Haversine distance was tried as a loss function but it did not have a good effect on the training of the model and was therefore abandoned.

$$d_{Haversine} = 2Rarcsin\sqrt{sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + cos(\varphi_1)cos(\varphi_2)sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)} \tag{2.5}$$

Equirectangular distance is a third loss function which was used in the Porto paper since their model did not work well with the Haversine distance loss function[9]. The mathematical formula used to calculate the equirectangular distance can be seen in equation (2.6). One of the perks of using the equirectangular distance formula is that it increases the computational efficiency of the training since it uses less resource intensive mathematical functions[41]. The equirectangular distance formula uses one trigonometric function and one square root function while the Haversine distance formula use seven trigonometric functions (four sines, two cosines, and one arcsine) and one square root function. The equirectangular distance is the loss function that was used to train the final model. The variables used in equation (2.5) and equation (2.6) are defined below.

$$d_{equirectangular} = R\sqrt{(\varphi_2 - \varphi_1)^2 + \left((\lambda_2 - \lambda_1)\cos\left(\frac{\varphi_1 + \varphi_2}{2}\right)\right)^2} \tag{2.6}$$

R is the radius of the earth (approximately 6,371 kilometers)

$\boldsymbol{\varphi_1}$ is the latitude of the first point measured in radians

$\boldsymbol{\varphi_2}$ is the latitude of the second point measured in radians

$\boldsymbol{\lambda_1}$ is the longitude of the first point measured in radians

$\boldsymbol{\lambda_2}$ is the longitude of the second point measured in radians

Optimizers

If we use the cost function to evaluate the performance of a machine learning model, then the objective is usually to either maximize or minimize the result of the cost function. In order to do this, optimizers are used and one of the most common optimizers is called gradient descent. In simple terms what gradient descent does is that it follows the gradient of the function downwards until it has reached the minimum value of the function. One important component of the gradient descent algorithm is the learning rate, the learning rate effects the size of the step in the intended direction[14].

One of the drawbacks of gradient descent is that it is slow. It can be accelerated by dividing the learning into batches, this is called stochastic gradient descent, which is the algorithm used in this thesis. Another component that can speed up the learning is the use of momentum. A pitfall of using any form of gradient descent is local minima which will cause the learning to converge to a halt even if it has not found the global minimum[14]. Even if it has some short comings, stochastic gradient descent remains a popular optimization algorithm within the machine learning field. The mathematical formulation for stochastic gradient descent can be seen in equation (2.7)

$$w^{(\tau+1)} = w^{\tau} - \eta \nabla E_n(w^{\tau}) \tag{2.7}$$

An alternative to the stochastic gradient decent algorithm is a relatively new method called Adam. Adam has shown good results when applied to big datasets with both big and small sets of features[22]. The purpose of Adam was to combine the perks of two other optimization algorithms namely RMSprop and AdaGrad.

Root mean square prop or RMSprop is an optimization algorithm which has proved to be an effective algorithm for deep neural networks and that is similar to Adam [22]. Unfortunately there exist no scientific publications about RMSprop however it is considered a well known algorithm within the machine learning field. RMSprop is an extension of Rprop but has included the use of mini-batches in order to make it more effective on large redundant datasets[38]. RMSprop modifies AdaGrad to make it better in a nonconvex setting by introducing a hyperparameter ρ. While AdaGrad performs well in a convex environment it performs worse in a nonconvex environment, RMSprop solves this issue by discarding history from extreme pasts which allows the algorithm to converge quickly once it has found a convex area in the nonconvex environment[14].

$$v(w,t) = \rho v(w,t-1) + (1-\rho)(\nabla E_n(w))^2 \tag{2.8}$$

$$w^{\tau+1} = w^{\tau} - \frac{\eta}{\sqrt{v(w^{\tau},t)} + \epsilon} \nabla E_n(w^{\tau}) \tag{2.9}$$

The mathematical formulation for RMSprop can be seen in equation (2.9). Note that the learning rate η is divided by the accumulated squared gradient which is calculated in equation (2.8). An explanation of the parameters in RMSprop and stochastic gradient descent can be seen below.

ρ is a constant which represents the decay rate

η is a constant which represents the learning rate

ϵ is a small constant which is added in order to avoid division by zero

2.5 Neural Networks

The term neural networks was coined as far back as 1943 but have since then disappeared and reappeared several times over the years[25]. Neural networks have become popular within

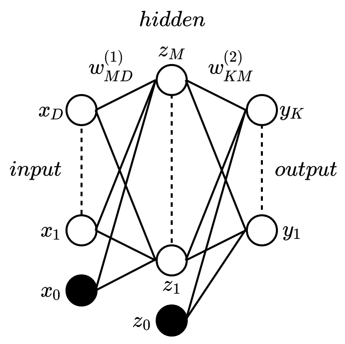


Figure 2.3: Multilayer perceptron (Source: [2])

the machine learning community as of late due to the fact that data is more easily accessible and gathered in bigger quantities. Neural networks are more complex and require more data than the conventional machine learning methods, however in many cases the results have been positive[37]. Neural networks are built up of different layers, the most simple model consist of one input layer, one output layer and a so called hidden layer. Each layer consists of an arbitrary number of neurons as designed by the developer.

In general the more layers and the larger amount of neurons that a model has, the better the model will be at solving complex tasks to a greater extent. However the model will be prone to overfitting. The model will also be computationally more expensive compared to a smaller model. The drawback of having a smaller neural network is that it is fallible to underfitting.

As with everything there are different approaches to neural networks however the method that has seen the most success in regards to pattern finding is called multilayer perceptron[2]. What this means in practice is that a single neuron in one layer is linked to all the neurons in the next layer. The technique is illustrated in figure 2.3.

A multilayer perceptron is sometimes called a feedforward neural networks. The term feedforward refers to the fact that information moves in one direction. The feedforward networks is a foundation for more complex architectures such as the convolutional neural network and the recurrent neural network. The former has proven to be useful when working with image recognition tasks and the latter is used in natural language processing[14].

One important component of a neural network is the activation function. The activation function is used to calculate the hidden units in the neural network. Three different activation functions are used in this thesis namely rectified linear units (ReLU), the hyperbolic tangent (tanh), and softmax.

The ReLU acitvation function has gained a lot of attention within the deep learning community after showing positive results in the Imagenet competition[17]. The rectified linear unit has shown slightly better error results when compared with the sigmoid function which is a classic activation function[12]. The equation for the ReLU can be seen in equation (2.10). The equation for the hyperbolic tangent can be seen in equation (2.11).

$$g(z) = max\{0, z\} \tag{2.10}$$

$$g(z) = tanh(z) \tag{2.11}$$

The softmax activation function is commonly used in classification models in order to generate a probability distribution over a set of classes[14]. Equation (2.12) is the mathematical representation of the softmax function. All the elements in the probability vector generated by the softmax function must be between zero and one and all the elements sum to one. In this thesis the softmax function is used to generate a probability vector over all the centroids generated by the K-Means algorithm.

$$softmax(z)_i = \frac{exp(z_i)}{\sum_j exp(z_j)} \tag{2.12}$$

It is possible to represent neural networks using mathematical formulations. Basically they can be seen as a series of functional transformations where each layer represents a functional transformation using an activation function such as those described above[2]. Figure 2.3 can be used as and example when illustrating how forward propagation works, which is the technical term for how information flows through a neural network. The first step involves calculating what is called activations. The first activations in figure 2.3 would be calculated using equation (2.13) assuming that the input layer is of size D.

Each activation is a linear combination of the output of the previous layer which in this case is x_i and a weight $w_{ji}^{(1)}$ where the subscript represents which layer the weight belongs to. In figure 2.3 the inputs x_0 and z_0 has been added and they are both permanently equal to one in order to absorb the bias terms $w_{j0}^{(1)}$ and $w_{k0}^{(2)}$ into the linear combination[2]. The activation is calculated for each neuron in the hidden layer and is then used as an input to a activation function. This step can be seen in equation (2.14) where $h(\cdot)$ is the activation function and z_j is called the hidden units which can be seen in figure 2.3.

$$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i \tag{2.13}$$

$$z_j = h(a_j) \tag{2.14}$$

z_j is then used to calculate the next set of activations using equation (2.15) where the size of M represents the number of neurons in the hidden layer. Notice that a different set of weights are used to calculate the new activations. The activations are then used as input to an activation function as is illustrated in equation (2.16). The result of the activation function will be the output of the neural network for the kth output unit[2].

$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j \tag{2.15}$$

$$y_k = h(a_k) \tag{2.16}$$

Equation (2.17) describes the model from the input layer to the output layer and can be thought of as forward propagation[2]. Note also that the activation functions $h(\cdot)$ in equation (2.17) is often chosen to be a sigmoidal function however the activation functions used in this thesis is the ones discussed above namely softmax, tanh, and ReLU.

$$y(x, w) = h\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right) \tag{2.17}$$

Once there is a complete mathematical representation of the model it needs to be trained so that the network parameters such as the weights can be set to optimal values. In order to do this a dataset of training examples is needed. If we have a dataset with N training examples divided into inputs $X = \{x_1, ..., x_n\}$ and expected outputs $T = \{t_1, ..., t_n\}$. One training example consists of an input vector x_n and an expected output value t_n. Once a training example has been executed by the model we need to find out whether the model was good or bad.

A way of evaluating the performance of the model during training is to use a loss function or cost function which was discussed in section 2.4. The loss function used in the final model of this thesis is called the equirectangular distance which calculates the distance between two geographical positions. In the context of taxi destination prediction the objective of the model training is to minimize the distance between the predicted destination and the expected destination. This could be done by minimizing equation (2.18) by finding the optimal weight vector w. The formula for $d_{equirectangular}$ can be seen in equation (2.6).

$$E(w) = d_{equirectangular}(y(x_n, w), t_n) \tag{2.18}$$

If we were to plot the loss function $E(w)$ we would get a surface and in the surface we would be able to observe a global minimum for a certain set of weights and perhaps some local minima for other set of weights. In order to construct the perfect model one would like to find the weights which results in the global minimum. In reality this is difficult and a local minimum might not be so bad if several different local minima have been evaluated[2].

In order to increase the speed of minimizing the error function it is common to use a optimization algorithm which were discussed in section 2.4. In this thesis RMSprop is used which is a form of stochastic gradient descent. Stochastic gradient descent updates the weights according to equation (2.7) where η is the learning rate. The weights move small steps towards the negative gradient which allows the error function to descend into local minima[2].

$$\delta_k = y_k - t_k \tag{2.19}$$

$$\delta_j = h'(a_j) \sum w_{kj}\delta_k \tag{2.20}$$

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \tag{2.21}$$

Backpropagation is a technique that is used to efficiently evaluate the gradient of the loss function. The following list explains the process of backpropagation[2]. In the example below equation (2.4) divided by two is used as the loss function.

1. Forward propagate information through the network as described above.
2. Evaluate δ_k for all output units using equation (2.19).
3. Backpropagate the δ's using equation (2.20) in order to retrieve δ_j for each hidden unit in the neural network.
4. Use equation (2.21) to evaluate the obtained derivatives.

Deep Learning

It can be very difficult to specify a feature vector and often a person with explicit expertise within the problem area is needed in order to figure out which features are more important than others. This is where deep learning comes into play. Deep learning makes it possible to enter raw data into the model which in turn will find complex patterns in the data by using multiple layers of representation. The discovered patterns will then enable the algorithm to create internal descriptions of the feature vector[23].

Figure 2.4 tries to illustrate the relationship between the areas machine learning, neural networks, and deep learning. Deep learning is a small area within neural networks, in turn neural networks is a small area within machine learning.

There does not exist any distinct definition of when an algorithm is considered to be a deep learning algorithm or a shallow learning algorithm. A vague but sensible definition is that a deep learning algorithm consist of several layers. An example of the potential in deep

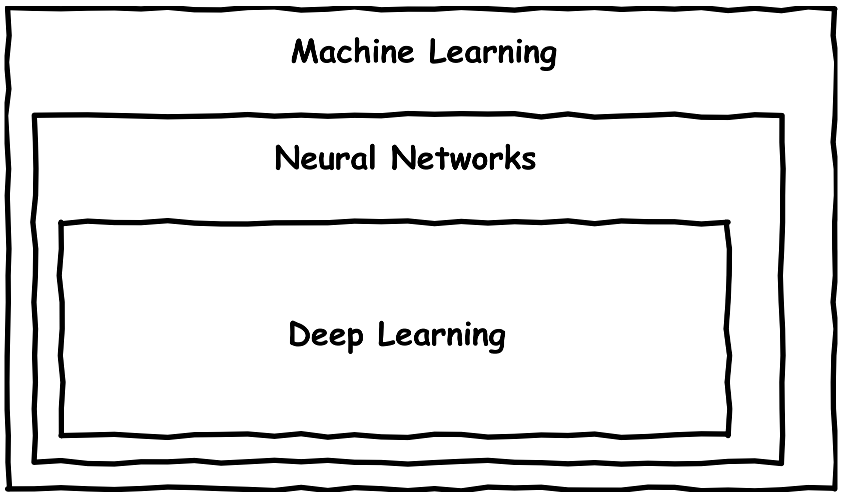


Figure 2.4: The relationship between artificial intelligence, machine learning, and deep learning (Source: [14])

learning models is depicted in the referenced literature. A deep learning architecture built up of 5-20 layers is able to distinguish a Samoyed from a white wolf[23].

To newly arrived practitioners of data mining, deep learning might seem like a modern technology. However the first occurrence of deep learning is dated back to the 1940's where it was called cybernetics. It was during this time that the first neuron model was described mathematically[25]. In the 1950's the single layer perceptron model was developed[32]. Then after laying dormant for around twenty years, deep learning resurfaced under the name connectionism in the 1980's. In this period the multilayered perceptron and the backpropagation algorithm were invented[33][42]. After another ten years of hibernation finally in 2006 the research field called deep learning bloomed[14]. The usefulness of deep learning became evident in step with the big data revolution.

2.6 Software Libraries

There exist several different tools which allow you to implement machine learning models. Ranging from low level libraries such as Theano and TensorFlow to high level APIs such as Keras and PyTorch. TensorFlow is an open-source software library for machine intelligence, it was originally developed by researchers at the Google Brain Team. One of the primary strengths of TensorFlow is its flexibility in regards of deploy-ability, it is easily deployed to cloud, desktop, or mobile devices using either CPU or GPU for computations[1].

TensorFlow and Theano can prove to be cumbersome when experimenting with different solutions which evolve rapidly. Due to this the technology of choice in this thesis is to use Keras as an initial prototyping tool in order to produce fast results[21]. One of the major perks of using Keras is that it is built upon several different low-level libraries such as Theano and TensorFlow. Once a working model has been built in Keras it can be exported as a TensorFlow model as well. Keras was chosen over PyTorch because PyTorch is a relatively new and still in beta while Keras 1.0 was released in April 2016 and as such it has been rigorously tested by the machine learning community[30].

Before using Keras you have to specify whether to use Theano, TensorFlow or CNTK as underlying architecture. In this study we have chosen to work with TensorFlow since it is built and maintained by Google, it is open source, and it has support for Python. The TensorFlow model can be used on the Google Cloud Platform or directly on an Android device. The second alternative would be to use Theano but recently the team behind Theano said that they will stop their work and focus on other things. Theano also lacks the mobile support which TensorFlow has.

Several other software libraries have been used for data manipulation, data visualization, and the clustering. Pandas is a open source Python library which provides convenient data structures and data analysis tools.[1] Pandas was used during this thesis to manipulate the dataset. Another open source Python library called scikit-learn which focuses on machine learning tools was used to transform certain features.[2] The K-Means algorithm provided in the scikit-learn library was also used in this thesis. Two other open source Python libraries called matplotlib and seaborn were used in order to visualize the data which provided a greater understanding of its content.[3,4]

2.7 Evaluation Techniques

In order to make sure that the extracted results seem reasonable and to be able to evaluate and compare different algorithms, an evaluation technique needs to be chosen. There exist different evaluation techniques depending on if you are trying to solve a classification problem or a regression problem.

In the case of regression a common performance metric is to compare the value calculated by the model with the expected value. In the case of taxi destination prediction the performance will be measured by calculating the equirectangular distance between the predicted coordinates and the expected coordinates. The mean equirectangular distance for all the taxi rides will then be used as a result of how well the model performs. The mathematical formulation for the mean equirectangular distance can be seen in equation (2.22).

$$\bar{d}_{equirectangular} = \frac{1}{n}\left(\sum_{i=1}^{n} d_{equirectangular(i)}\right) \tag{2.22}$$

When evaluating classification algorithms two popular evaluation metric is precision and recall which can be combined into a single metric called F-score or F-measure. A core component in the precision and recall methodology is the confusion matrix which is illustrated in figure 2.5. Imagine that an algorithm can predict either a 1 (positive) or a 0 (negative), and the actual value can be either a 1 (positive) or a 0 (negative). If the predicted value is the same as the actual value, we say that the result is true. If the predicted value differs from the actual value, then the result is false. The second part of the result is either negative or positive depending on the predicted value. A concise definition follow below.

True positive is when both the predicted and the actual value is 1 and thus a correct result

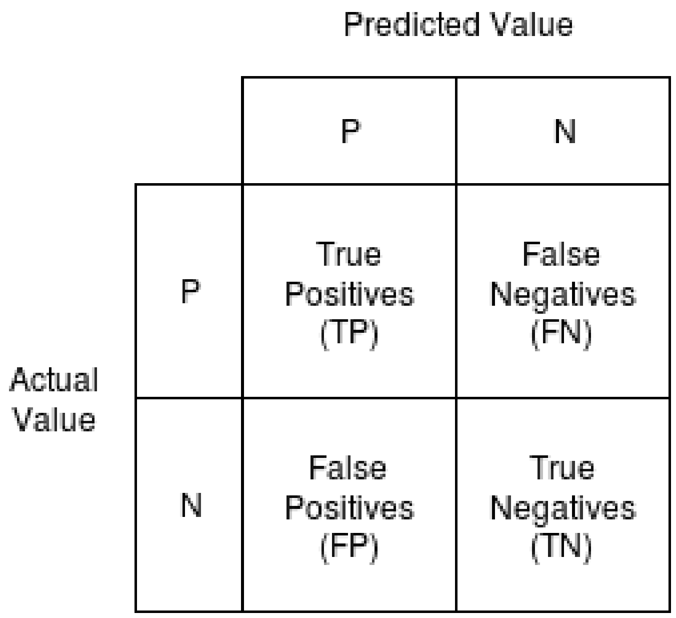
True negative is when both the predicted and the actual value is 0 and thus a correct result

False negative is when the predicted value is 0 and the actual value is 1 and thus an incorrect result

False positive is when the predicted value is 1 and the actual value is 0 and thus and incorrect result

The formula for precision is displayed in equation (2.23) and the formula for recall can be seen in equation (2.24). Precision is the fraction of actual positive predictions divided by the total number of positive predictions. In other words we get a number which says how many of the positive predictions were actually correct. Recall is the fraction of actual positive predictions divided by the number of true positive and false negative predictions. Recall is a measure of how many of the actual positive results were classified as being positive [39].

[1]http://pandas.pydata.org/
[2]http://scikit-learn.org/stable/
[3]https://matplotlib.org/
[4]https://seaborn.pydata.org/



		Predicted Value	
		P	N
Actual Value	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Figure 2.5: Confusion matrix (Source: [27])

$$Precision = \frac{truepositive}{truepositive + falsepositive} \tag{2.23}$$

$$Recall = \frac{truepositive}{truepositive + falsenegative} \tag{2.24}$$

Precision and recall can then be used to calculate the F-score as seen in equation (2.25). The purpose of F-score is to combine precision and recall into one single metric[39].

$$Fscore = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.25}$$

Equation (2.26) is a measure of the global accuracy of the predictions. The total number of true predictions is divided by the total number of predictions which results in a factor that represents the accuracy[27]. If the result is 1 then the model has 100 percent accuracy while if the result is 0.5 then the model has 50 percent accuracy.

$$Accuracy = \frac{truepositive + truenegative}{truepositive + truenegative + falsepositive + falsenegative} \tag{2.26}$$

2.8 Recommender Systems

Recommender systems provided inspiration for how to solve the problem of predicting a taxi destination from a technical point of view. It is possible to say that the first model that was constructed in this thesis uses a form of collaborative filtering. The recommender system pipeline is a technique which is similar to what has been done in this thesis, even if it is on a very small scale in this thesis.

The core functionality of a recommender system is to predict an object that is worthwhile to recommend to the user, where an object can be anything from a song or a taxi ride destination. One type of recommender system is the content-based system in which a user receives recommendations depending on what the user has liked in the past. Each object that is likeable has a set of quantified features which can be compared between objects in order to determine the likelihood that the user will appreciate the object. If the likelihood is above a certain threshold then the item will be recommended to the user[31].

One of the most widely used techniques is called collaborative filtering which works by recommending items that a user of similar taste has liked. A popular real life example is Netflix which uses collaborative filtering to recommend movies to its users, this will be discussed in greater detail later in this thesis. There is a method called knowledge-based systems, they work by building a use case and then matching the users needs with the recommendation. Then there is the hybrid recommendation system which combine the methods used in the specific systems mentioned above[31].

One of the problems with constructing a recommender system is that there is many different information sources and many different parameters that needs to be taken into account before a recommendation is given as an output. An approach to building recommender systems is to build a so called recommendation pipeline. This allows for systematic processing of the data and makes it possible to digest several sources of information and then bundle them together for a final analysis. In the first stage of the pipeline training data is generated from user behaviour for example. In the second stage a model is trained and verified[3].

In a paper published by Google engineers where they use deep neural networks for YouTube recommendations, they used an interesting recommendation system architecture[7]. Basically they used two different neural networks. The first network filters all available YouTube videos down to a couple of hundred videos depending on what kind of videos are relevant to the user, this is called candidate generation. Then the second network gives every movie in the subset a rating by comparing the user's features and the movies features. The movies with the highest rating then gets presented to the user[7].

Netflix

Netflix started a competition called the Netflix Prize which is a machine learning and data mining competition. The goal of the competition was to create a solution which performed 10% better at movie rating prediction compared with the current algorithm used at Netflix. The prize money was one million dollar and the competition helped with bringing a lot of attention to the area of recommendation systems. Two algorithms that really stood out among the first successful solutions was Matrix Factorization and Restricted Boltzman Machines[31].

2.9 Similar Work

This section contains a summary of the taxi destination prediction challenge and a paper from the winners of the competition, which will be referred to as the Porto paper in this thesis. The destination prediction system used by Uber will also be reviewed and the Spotify discover weekly feature has also been looked at.

Taxi Destination Prediction Challenge

In 2015 a taxi destination prediction competition was organized by ECML/PKDD (the European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases) which urged attendees to solve a problem that in some ways is similar to the problem that this thesis attempts to solve. The objective was to predict the destination of the cab ride with help of data that contained information about cab rides in Porto (Portugal). The dataset consisted of 1.7 million taxi rides withholding GPS (Global Positioning System) positions and meta data such as client ID, taxi ID, and time stamps[9].

In the winning solution they used an artificial neural network with a multilayered perceptron architecture which was discussed in greater detail in section 2.5 of this thesis. The developers of the winning model tried different solutions and experimented with recurrent neural networks and bidirectional recurrent neural networks which also showed promising

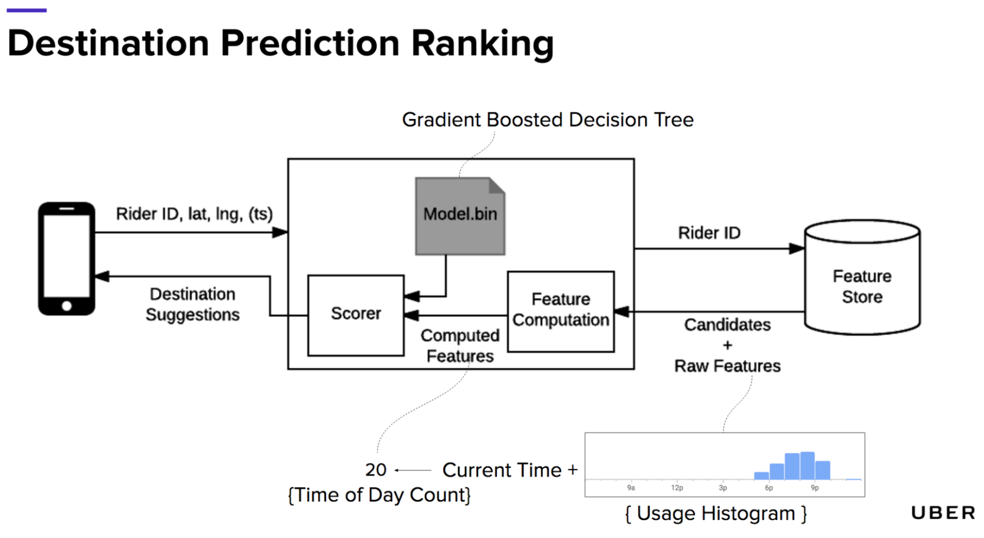


Figure 2.6: Uber destination prediction system (Source: [40])

results. The results and methods in the paper published from the winners of the competition has been influential in the choices of techniques deployed in this report. The conclusion that was brought from the paper was that neural networks were good at predicting taxi travel destinations from the data provided in the challenge and thus neural networks seems like a good approach to the problem of this thesis.

Uber Destination Prediction

The global transportation technology company Uber released a feature that is called Destination Prediction in November 2016. It is a good real life example that shows how a taxi destination prediction system can be constructed and that it is a tool which the users appreciate. 50 percent of all destination entries are done through the destination predictor. The feature set that is provided by the user in the Uber case is similar to the information that was found in the Taxi Stockholm dataset. The features are a rider identification, a latitude coordinate, a longitude coordinate, and a time stamp which is illustrated in figure 2.6[40].

The system also uses historic user data in order to get better predictions that resemble how you usually travel. Taxi Stockholm has the majority of its business in Stockholm, but Uber has users in over 600 cities which means that you probably do not have personal user data for each new city that you visit. Uber solves this by using travel patterns from other users and try to identify locations that are usually traveled to. They use a technique which they call the donut, this means that they look at an area between 400 feet and 400 miles for previous user history or points of interest. Anything outside the range of the donut is considered irrelevant since you are unlikely to take a ride that is shorter than 400 feet or longer than 400 miles[40].

Spotify Discover Weekly

According to a blogpost which has gathered a lot of public knowledge about the Spotify Discovery feature, Spotify uses three different sources of information to feed its recommendation pipeline[5]. The first source is a collaborative filtering of all the music consumed by the users of Spotify. A simplification of how it works is that a user receive recommendations from people with similar music taste. If you have listened to a lot of similar tracks as another user, then you will receive recommendations of other tracks that that user has listened to that you have not listened to. This goes both ways so the other user will receive recommendations depending on what you have listened to[5].

The second system uses a technique called natural language processing. What this system does is that it crawls the web for descriptions of songs. Then it compares the adjectives used to describe a song with the description of another song in order to find similarities. The third system uses convolutional neural networks to analyze the raw audio data of a song. The result of the convolutional neural network is song characteristics such as tempo and loudness which then can be used to compare songs and find new tunes with similar traits[5].

3 Method

This thesis has followed a knowledge discovery process called KDD which was presented in section 2.1. This chapter reproduces all the steps conducted during this thesis project and have been divided into the following sections.

- Data Analysis
- Data Mining
- Evaluation

The data analysis section shows what has been done during the first three steps of the KDD process namely, data selection, data cleaning, and data transformation. The data mining section shows the neural network model and the tuning process of the model. The last section talks briefly about how the model was evaluated during the different stages of the project.

3.1 Data Analysis

The first step in the KDD process is called data selection and it is the first step in this thesis methodology as well. The first important decision was to decide which dataset should be used, what data from the dataset should be included, and what additional data should be added.

The different available datasets were presented in section 2.2 and it was decided to work with the Taxi Stockholm dataset. It was Taxi Stockholm that wanted to evaluate the possibility of using destination predictions in their mobile application and because of this it made sense to use their dataset. The dataset was delivered as a comma separated file and the first task was to analyze the content of the data which can be seen in table 2.1.

In order to use the departure and destination features the addresses had to be converted from strings to a discrete value. Geocoding was used in order to convert the addresses into coordinates and Bing Maps was the service of choice for this task. Bing Maps was the service that had the best balance between coordinate accuracy and the number of free data conversions per day. Once the conversion was complete the coordinates were added to the dataset.

Another part of the data selection phase is to remove some of the features which were deemed unnecessary. More precisely the from_zone, to_zone, passenger, and phone features were removed. The motivation behind the removal of the from_zone and the to_zone was that all rides did not have a specific zone assigned to it. In some cases a taxi ride had arrived at the same destination but only one of the rides had a to_zone, so the feature was considered inconclusive and thus removed. Every unique user had a unique passenger name and a unique phone number. As such the features served the same purpose as the user identification and were removed because they seemed superfluous.

The final dataset contains the coordinates of both the departure and the destination, and the original addresses were also kept in order to enable validation of the correctness of the coordinates. The date and time of the ride was also saved together with the booking device type and the booking type.

Once the data had been selected it was time to clean the data. Some of the rides did not have a user identification, some rides missed a departure address or a destination address. The first step in the cleaning process involved the removal of faulty rides. The motivation for their removal of the faulty rides was that they had a negative impact on the model and in some cases ruined the model completely by inputting null values.

The geocoding process was not flawless and some of the taxi rides got coordinates that were located outside of Stockholm and even outside of Sweden. A second cleaning of the data was done by limiting the geographical area in which a taxi ride had to be in for it to be included in the dataset. The area was limited to a latitude between 58 and 60 and a longitude between 16 and 19. This removed some rides which had departure or destination locations outside of Sweden, those rides had a big negative impact on the result of the model. The motivation behind the chosen geographical area is that it removed many rides that were far outside of Stockholm. However the area was still large enough to include some important geographical locations such as the Arlanda airport and the industrial city Södertälje.

The original dataset consisted of exactly 677,134 taxi rides and once all the cleaning processes were completed, exactly 295,940 taxi rides remained.

Feature	**Description**
Latitude	A floating-point number which represents the standardized departure location on the latitude axis.
Longitude	A floating-point number which represents the standardized departure location on the longitude axis.
Time	A binary representation (1 = True, 0 = False) of the time of the day divided into the following seven categories: **Morning** - true if the time is between 05:01 and 08:00 **Forenoon** - true if the time is between 08:01 and 11:00 **Lunch** - true if the time is between 11:01 and 13:00 **Afternoon** - true if the time is between 13:01 and 16:00 **Home** - true if the time is between 16:01 and 19:00 **Afterwork** - true if the time is between 19:01 and 22:00 **Night life** - true if the time is between 22:01 and 05:00
Weekday	A binary representation (1 = True, 0 = False) of the type of day divided into the following four categories: **Monday** - true if the day is Monday **Midweek** - true if the day is Tuesday, Wednesday or Thursday **Friday** - true if the day is Friday **Weekend** - true if the day is Saturday or Sunday

Holiday	A binary representation (1 = True, 0 = False) of the type of week divided into the following five categories: **Winter sports holiday** - true if it is week 9 **Easter** - true if it is week 15, which is Swedish Easter 2017 **Summer** - true if the week is between 27 and 33 **Autumn** - true if it is week 44 **Regular week** - true for every regular week (none of the above)
Passenger type	A binary representation (1 = True, 0 = False) of the type of user divided into the following categories: **First time user** - true if the user is new to the service **Second time user** - true if the user has used the service once before **Returning user** - true if it is the users third ride **Regular user** - true if the user has taken between four and ten rides **Frequent user** - true if the user has taken more then ten rides
Booking device	A binary representation (1 = True, 0 = False) of the type of device used in the booking, divided into the following three categories: **iPhone** - true if an iPhone was used **Android device** - true if an Android device was used **Website** - true if the website was used
Booking type	A binary representation of the type of booking, 1 represents a pre-booked ride and 0 represents a directly booked ride
User identification	A floating-point number between zero and one which represents a unique user identification

Table 3.1: The final feature set

Once the street addresses have been converted to coordinates, the coordinates are normalized by removing the mean and scaling to unit variance. This is done with the help of the StandardScaler function which is found in scikit-learn, which is a machine learning library for Python.[1] The StandardScaler function does the same things as the zero-mean normalization which was presented in section 2.3 and can be seen in equation (2.1).

The day of the week feature, the week of the year feature, and the time feature were aggregated into a binary representation using the 1-of-K scheme which was presented in section 2.3. The day of the week feature was transformed from a discrete value into the states Monday, Weekday, Friday, and Weekend. The week of the year feature was transformed from a discrete value into the states Winter sports holiday, Easter, Summer, Autumn, and Regular week. The time feature was also divided into the states Morning, Forenoon, Lunch, Afternoon, Home, Afterwork, and Night life. The complete list of features can be seen in table 3.1

[1]http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing

The booking type feature was aggregated into a binary number which says whether the taxi ride was pre-booked or directly booked. The booking device feature was aggregated using the 1-of-K scheme with the categories being iPhone, Android device, or Website. The type of passenger also seemed like an interesting feature, by looking at the number of rides done by a passenger they were divided into five different categories using the 1-of-K scheme. The first time user, the second time user, the returning user, the regular user, and the frequent user. The contents of the passenger type feature can be seen in table 3.1.

The final aggregation was the user identification which was transformed to a floating-point number in the range zero to one. This was accomplished using a function called MinMaxScaler which is found in the scikit-learn library.[2] The MinMaxScaler function does the same things as min-max normalization which was presented in 2.3 and can be seen in equation (2.2). The minimum value that was used was zero and the maximum value was one, so the transformed data ranged between zero and one.

The methodology for dividing these features into categories was to look for patterns in the graphs of the data. It was decided to use the 1-of-K scheme because of two reasons, it is a comprehensible way of representing human behaviour and it creates an equilibrium between the features. The motivation behind using normalization on the user identification is that the normalized value is on the same scale as the binary representations. The reason for normalizing the coordinates is because it removes the measurement unit. All the techniques used in this section can be read about in greater detail in section 2.3 and in the discussion.

3.2 Data Mining

The data mining step of the KDD process involves choosing what type of data mining to use to achieve the goal of the process. The goal of the process is to give a prediction of where a taxi customer wants to go without entering the street address. To achieve that goal two different machine learning methods were used. The first method is a neural network which was used to calculate the destination prediction. Neural networks were chosen because they had shown positive results solving similar problems. Neural networks were used in the taxi destination prediction challenge and a lot of inspiration was taken from the winner of that competition.

The second method used in combination with the neural network is clustering. In the Porto paper a positive effect was noticed on the performance of the model when a vector containing cluster centroids of previous taxi rides was added. A centroid is the mean of a cluster. A similar effect was noticed during the early stages of model construction in this thesis and that is why they are included in the model. In the early stages of the thesis there was an hypothesis that the number of centroids could have a significant impact on the performance of the model. Because of this hypothesis the K-Means algorithm, which was brought up in section 2.4, was chosen to generate the clusters.

It is easy to specify the exact number of centroids which should be generated from the algorithm. Because of this it was simple to do a grid search and evaluate many models with a different number of centroids. The cluster centroids are created using the K-Means clustering function from the scikit-learn library.[3]

In the final model which can be seen in figure 3.1 it is possible to see that the model starts with an input layer which is connected to a hidden layer. The first and the second hidden layer uses the rectified linear unit activation function and 100 neurons each. The third hidden layer uses the hyperbolic tangent activation function and has 50 neurons. The fourth hidden layer uses the softmax activation function which outputs a probability vector which corresponds to the probabilities of traveling to each centroid. The output of the model will be the weighted average of the probability vector produced by the softmax activation function

[2]http://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing
[3]http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

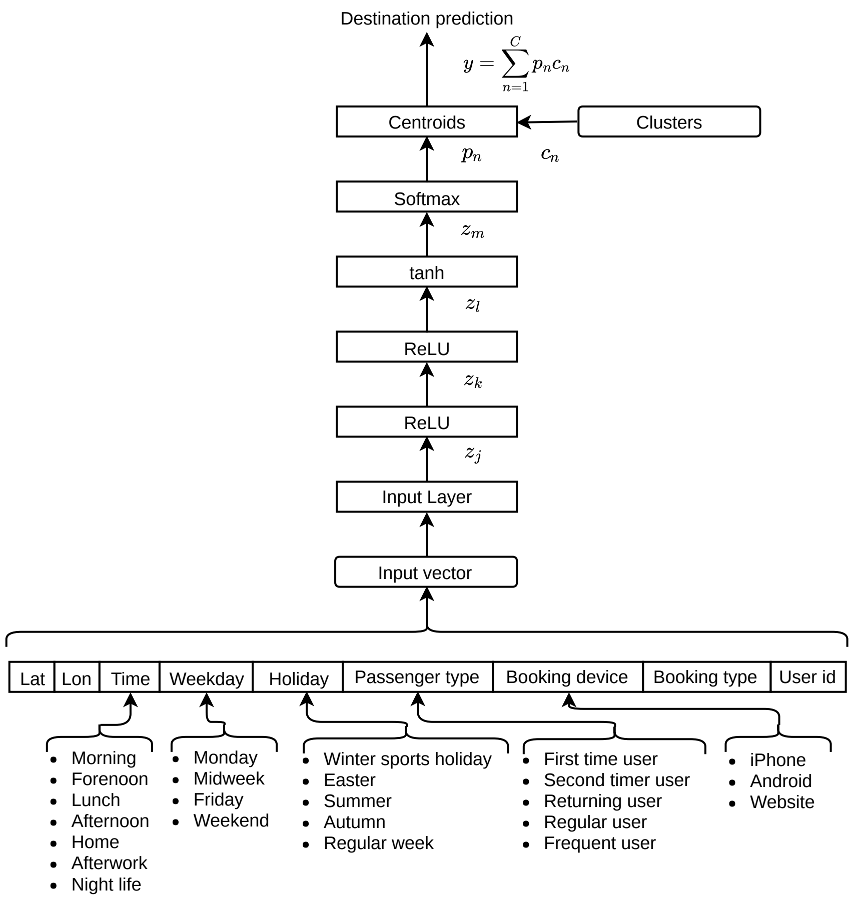


Figure 3.1: The final model

and the centroid vector produced by the K-Means algorithm. The result of the weighted average is a set of coordinates which is the predicted destination.

The ReLU activation functions are used because they are the recommended activation functions in most cases. The last layer before the softmax uses tanh as activation function and that is because the model produced bad results when there was three ReLU's in a row. The motivation behind using softmax to produce a probability vector and then use the centroids to calculate a weighted average is mostly taken from the Porto paper. Their solution to predict taxi destinations was analyzed and the conclusion was that it was a reasonable approach to solve the problem.

The number of layers and the size of the layers were determined by a hyperparameter tuning process using grid search. A minimum size was set to be a model with one layer consisting of ten neurons and the biggest model had three layers consisting of 2000, 2000, and 1000 neurons respectively. Several models in between the minimum and maximum size were trained on the training data and evaluated using the validation data in order to find an optimal model. It was decided to use model 2.3 in table 4.1.

The performance impact of the number of clusters was also evaluated in a similar fashion using grid search. The minimum cluster size was set to ten and the maximum cluster size was set to 2000 and several models between the minimum model and maximum model were evaluated. The result from the cluster size grid search can be seen in table 4.2. The maximum and minimum model sizes in both grid search evaluations where inspired by model sizes used in previous similar work.

The model uses a custom made loss function called equirectangular distance which calculates the distance between two coordinates, the formula can be seen in equation (2.6). The motivation for using the equirectangular distance formula is because it works better then mean squared error when calculating the distance between two coordinates. The optimization algorithm used to speed up the training process of the neural network is called RMSprop which is presented in greater detail in chapter 2.

The variables of RMSprop use the standard settings that is provided with the Keras library which is a learning rate of 0.001, ρ is equal to 0.9, ϵ is equal to 1e-08, and the decay is set to 0.0. The values of the RMSprop variables is the standard values and they have not been changed, the reason for this is that there simply was no time to experiment with different values and the standard settings had a satisfying performance. The motivation for using RMSprop is that it worked better then stochastic gradient descent and Adam during early experimentation with the model.

Once the model size was finalized it was time to personalize the model by editing the dataset. During the construction and evaluation of the model, a general dataset containing all the taxi rides was used. The hypothesis of this thesis is that the model can be improved for a user with many previous taxi rides by increasing the usage of personal data. The definition of personal data in this thesis is all the taxi rides that belong to a certain user, however the features are the same in both the personal data and the general data. The amount of personal data is increased by oversampling the personal data and undersampling the general data. By doing this the original size of the dataset is kept but the balance between personal and general information is altered.

The motivation behind the undersampling and oversampling is that the total number of taxi rides is 295,940, while the user with the most taxi rides has 457 taxi rides. In the case of the user who has 457 taxi rides, the personal data represents 0.15 percent of the total amount of data which means that the personal information will be drowned by the general information. By increasing the personal information and reducing the general information, the model will become tailored to a specific user and hopefully perform better for that specific user.

$$data = \alpha \cdot data_{general} + \beta \cdot data_{personal} \tag{3.1}$$

An in-depth discussion about different sampling techniques can be found in section 5.2, in this thesis the personal data will be oversampled by simply duplicating the existing taxi rides. This method of oversampling data could have been improved by adding small deviations in the data points, however due to time limitations this was not done. The undersampling of the general data was done by selecting random samples which is a technique for reducing datasets. We introduce the variable α which controls how much general data will be used and the variable β which controls how much personal data will be used. In figure 3.2 it is possible to see the five different levels of personalization which will be evaluated. The formula for the dataset construction can be seen in equation (3.1).

The mathematical formulation of the final neural network can be seen in equations (3.2)-(3.7). All the sums start from zero instead of one so that the bias term is absorbed into the sum. The value of C in (3.7) represents the number of centroids generated by the K-Means algorithm and during the evaluation of the α and β variables, $C = 25$ was used. The motivation for using 25 clusters is because the performance did not vary much between the cluster sizes and the fewer clusters that are generated, the fewer unique trips a user needs to have in

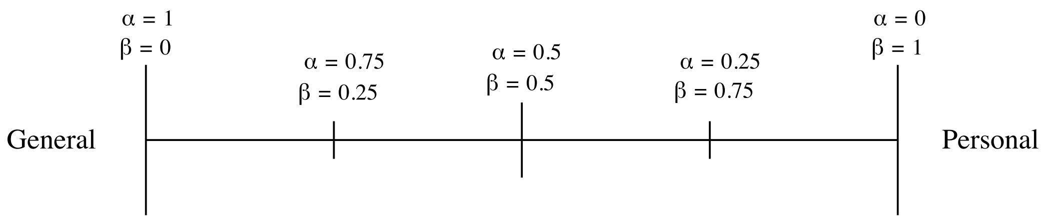


Figure 3.2: Division between general and personal data

order to have a fully personalized model. Because the K-Means algorithm needs a minimum amount of unique locations in order to produce a certain amount of clusters.

$$z_j = \sum_{i=0}^{28} w_{ji}^{(1)} x_i \tag{3.2}$$

$$z_k = ReLU\Bigg(\sum_{j=0}^{100} w_{kj}^{(2)} z_j\Bigg) \tag{3.3}$$

$$z_l = ReLU\Bigg(\sum_{k=0}^{100} w_{lk}^{(3)} z_k\Bigg) \tag{3.4}$$

$$z_m = tanh\Bigg(\sum_{l=0}^{100} w_{ml}^{(4)} z_l\Bigg) \tag{3.5}$$

$$p_n = softmax\Bigg(\sum_{m=0}^{50} w_{nm}^{(5)} z_m\Bigg) \tag{3.6}$$

$$y(x,w) = \sum_{n=1}^{C} p_n c_n \tag{3.7}$$

3.3 Evaluation

Evaluation of the results was done during several steps of the thesis. The quality of the data and the performance of the selected features were evaluated by running small experiments during the initial phase of the project. One important step before gathering results was to divide the dataset into three smaller sets of data using a 60/20/20 percent split. The training data consists of 60 percent of the total data, then there is a validation set which contains 20 percent of the rides and lastly a test set which also contains 20 percent of the taxi rides.

The motivation for how the dataset was split is because this is the method that Andrew Ng used during his Stanford course. The training dataset was used only when training the model, the validation data set was only used when evaluating the hyperparameters of the model and when evaluating the different datasets, and the test set was used to gather results which can be used to compare the models of this thesis with other models.

The first model evaluation was conducted during the tuning of the hyperparameters. First the number of layers and the size of the layers was decided by comparing the performance of different models on a validation set. The number of layers and the size of the layers were set to match the model which performed the best. The next step was to evaluate the optimal cluster size, which was done in a similar manner to how the size of the layers and the number of layers were evaluated. In the end these results were used as proof that the difference in cluster size did not impact the performance of the model in a big way. Even though the optimal cluster size according to the performance on the validation data was not chosen, it would not impair the model.

The evaluation of the hyperparameters was done by looking at the results of the validation loss of the model. Both results are the average error produced by the equirectangular distance. Keep in mind that the input to the loss function has been normalized so the result has no measurement unit and as such the results can be regarded simply as a score which can be compared between models. The lower score a model has, the better it performs since the objective of the neural network is to minimize the score.

After picking a model size and setting the cluster size to 25 in order to make sure that users with few previous rides could use a more personalized model. The same evaluation technique was applied when attempting to find the optimal blend of personal data and general data. Five different users were picked with a varying amount of previous taxi rides. Each user received five different training sets with varying degrees of personal and general data which were evaluated. All the users used the same neural network model but each model was trained with different datasets and consequently a different set of centroids. Each model was trained with a unique dataset generated from personal and general data according to the formula in (3.1) and the α and β in table 4.3.

A validation set was constructed using only personal data so that the result would mimic the use of the model in a real life scenario. The reason for this was so that the score of the five different models could be compared in order to find the best balance between personal and general information. The model that performed the best on the validation set for each user was then chosen to be compared to other machine learning methods.

A test set was also constructed from personal data in order to compare the final neural network model with a random forest model and a k-nearest neighbor model. This comparison was carried out in order to evaluate how well the neural network model performs compared to other techniques. The random forest algorithm from scikit-learn was used with a max depth of ten which after experimentation seemed to give the best results.[4] The k-nearest neighbor algorithm from scikit-learn was used with number of neighbors equal to four which seemed to give the best results after some experimentation.[5]

The random forest model and k-nearest neighbor model was trained using the same training data as the neural network which performed the best on the validation data. The neural network model which performed the best on the validation set for each user was then compared to the random forest model and the k-nearest neighbor model using the test data.

[4]http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
[5]http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

4 Results

This chapter presents all the results which were gathered during this thesis project beginning with the results of the data analysis which contains the data selection, data cleaning, and data transformation phases. Following the data analysis is a section about the tuning of the hyperparameters and lastly the results gathered from the personalized model.

4.1 Data Analysis

In the beginning of the thesis it was important to verify the validity of the dataset and to make sure that the data was correct. Many hours was put into visualization of data in order to verify the dataset. It is possible to see what looks like a coarse map of Stockholm in figure 4.1. It is possible to distinguish the different districts of the city such as Södermalm, Gamla Stan, Kungsholmen, and Östermalm. The image was produced by plotting each taxi ride destination point and departure point as a small white dot on a black background.

Another image which is zoomed in on the city of Stockholm can be viewed in figure 4.2. If the purpose of figure 4.1 was to illustrate that it is possible to paint a map of the different districts of Stockholm. Then the purpose of figure 4.2 is to show that it is possible to draw a map where it is possible to distinguish the individual streets. If you look at the district Södermalm it is possible to see some of the major streets such as Ringvägen and Götgatan.

The final geographical plot can be seen in figure 4.3 which is a heat map containing most of the taxi rides in the dataset. The black areas does not have any taxi rides in them and the color changes from a dark purple color into yellow depending on the number of rides within each square. The big yellow area in the middle is the central parts of Stockholm. Then there is some taxi activity in the suburbs of Stockholm while in the outskirts of Stockholm there is barely any taxi activity at all. It is possible to see that there is a small yellow square in the top left of the image as well and that square has the same geographical position as the Swedish airport Arlanda.

Results Of Data Analysis

After spending some time with the spatial data analysis it was time to take a look on the raw numbers in the data. In figure 4.4 the distributions of rides over the hours of a day have been plotted. During early morning hours there is a surge of taxi rides which declines abruptly after nine o'clock. Then there is a slow build up of taxi rides during the afternoon which once again dips around dinner time. Then there is a small rise in the evening before declining again in preparation for the morning surge.

In figure 4.5 the number of rides per weekday is illustrated. It is clear that Wednesday is the most traffic intense day tightly followed by Thursday and Tuesday. The workday with lowest amount of traffic is Monday, which has roughly 10,000 or 20 percent less rides compared to the busiest workday which is Wednesday. Fridays have slightly higher traffic as compared to Mondays but a slightly lower traffic when compared to the rest of the workdays. Then there is a substantial loss in traffic during the weekends starting with Saturday which has approximately 30,000 rides and then Sundays which has approximately 25,000

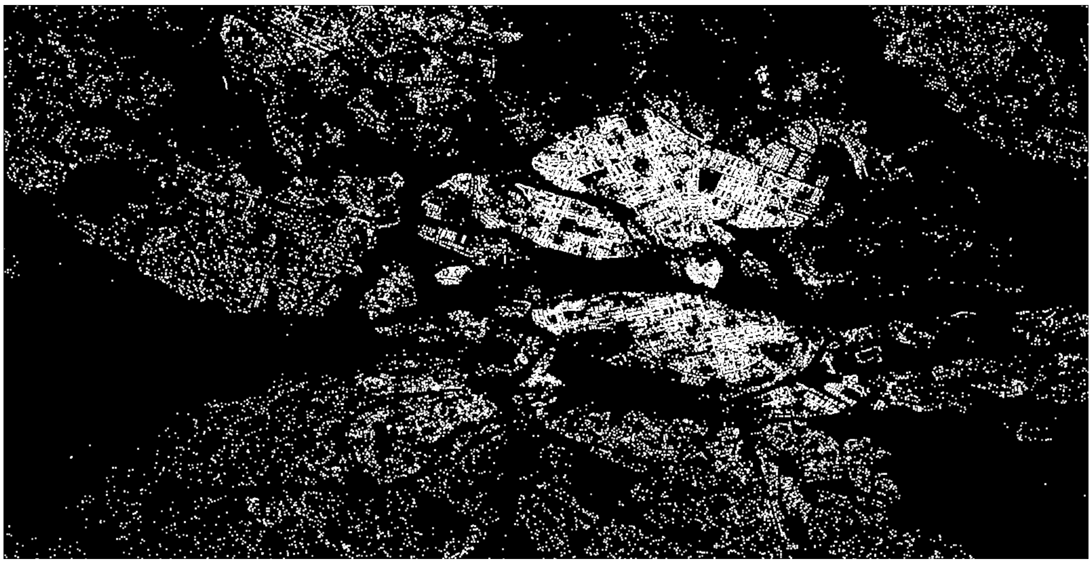


Figure 4.1: An overview of the taxi rides in Stockholm City

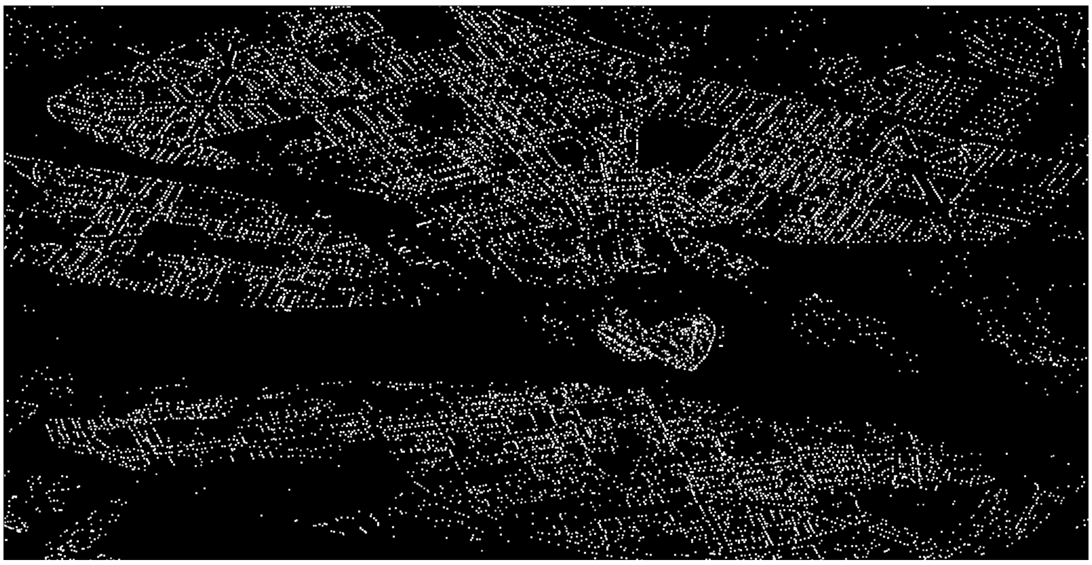


Figure 4.2: A fine detailed image of the taxi rides in Stockholm City

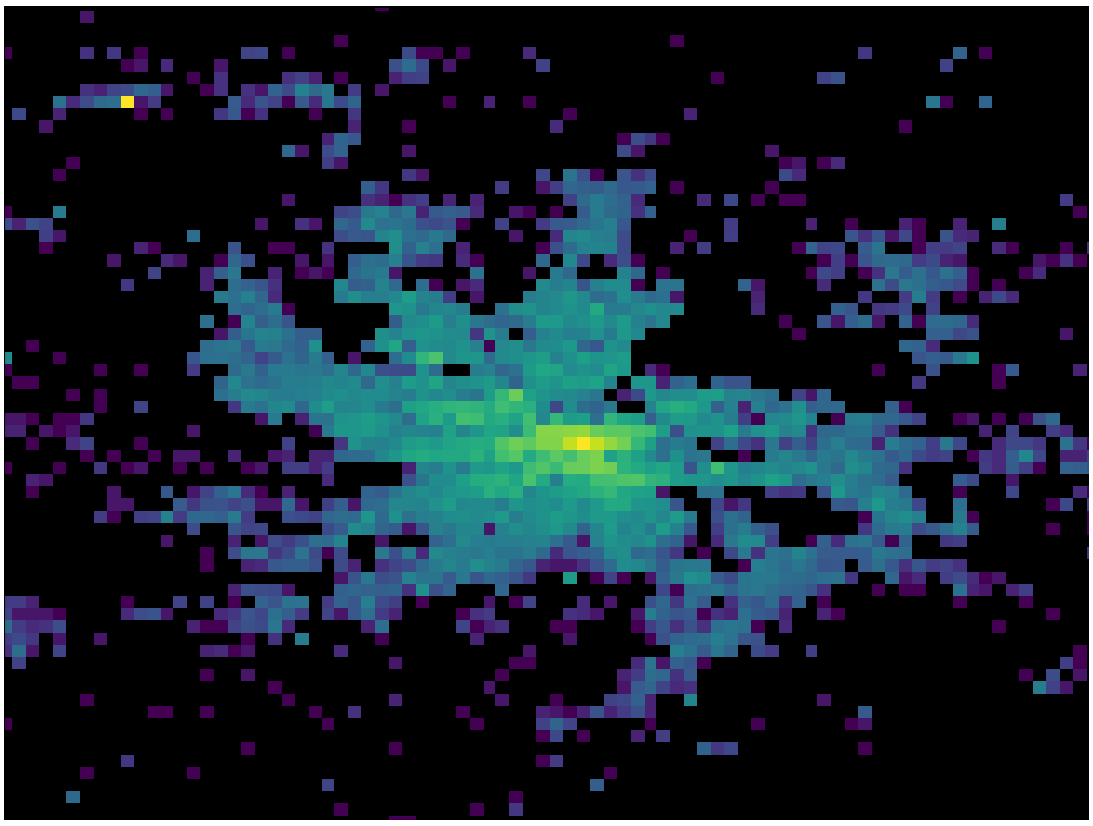


Figure 4.3: Heat map of the taxi rides around the city of Stockholm

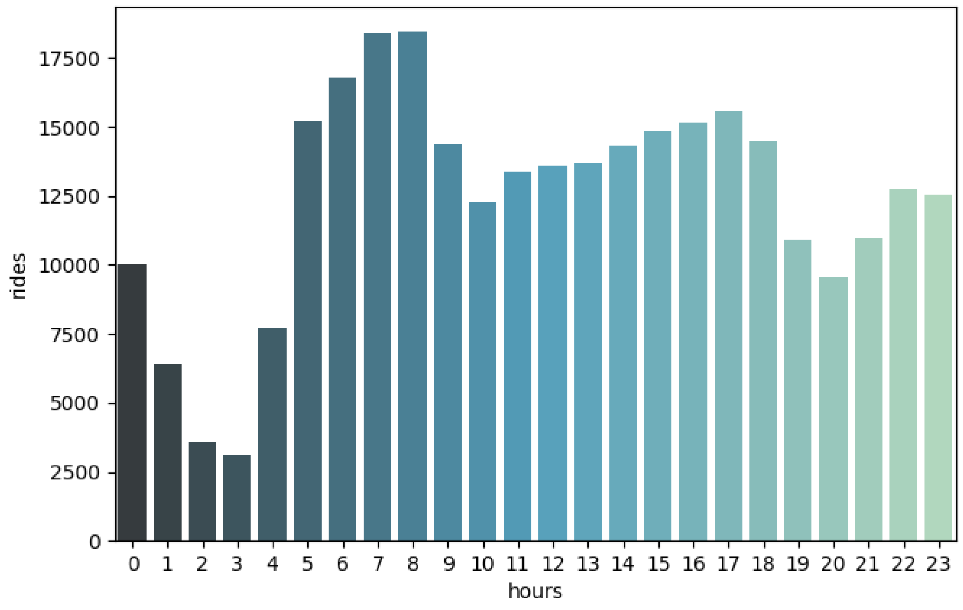


Figure 4.4: Rides per hour

rides. The accumulated rides of both Saturday and Sunday is only slightly bigger then the number of rides of the one day Wednesday.

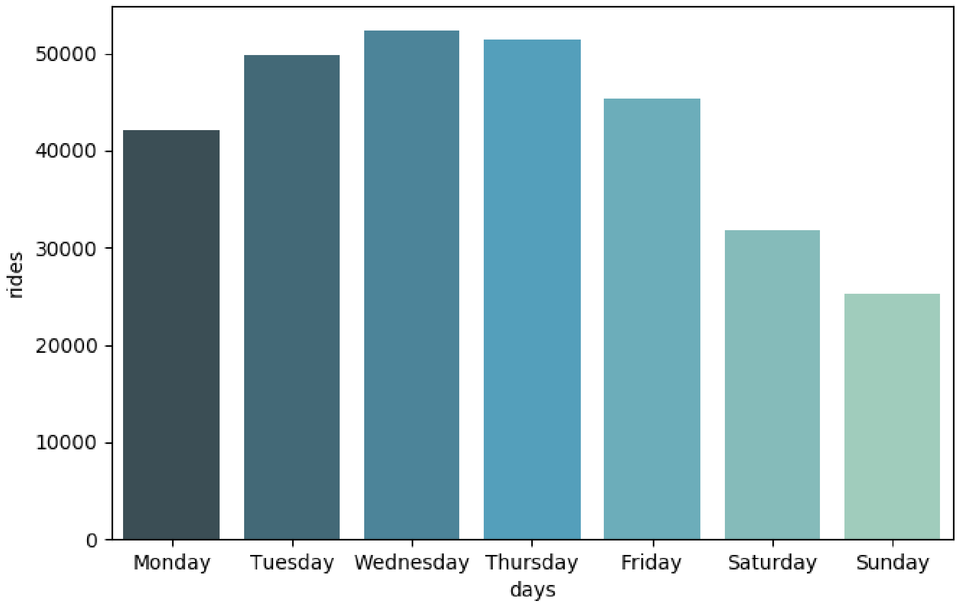


Figure 4.5: Rides per day

In figure 4.6 the number of rides per week is illustrated in order to show the distribution of rides over the year. One should keep in mind that the dataset is limited to an interval from week 9 to week 48 which can be seen in figure 4.6. There are some weeks that stand out from the rest. Week 9,15, and 21 have a lower ride frequency compared to most other weeks during the spring season. There is also an interval between approximately week 27 and week 33 with a low amount of taxi rides. Then during the later parts of the year there is two weeks which stick out, namely week 44 and week 48, where a decrease in traffic is noticed.

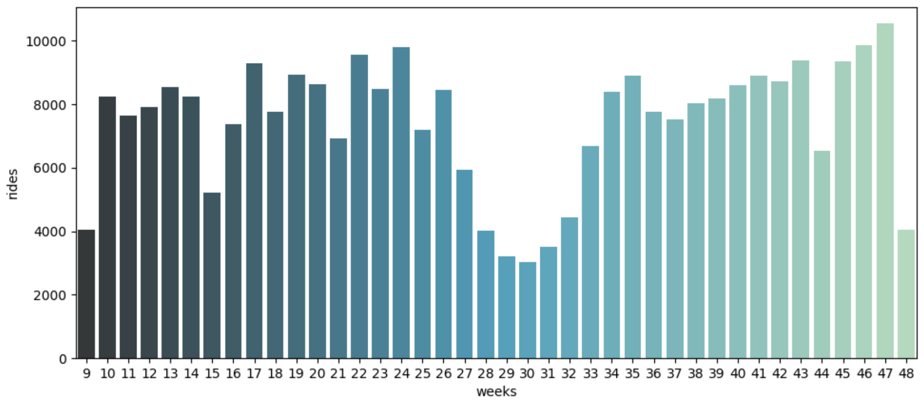


Figure 4.6: Rides per week

In figure 4.7 the distribution of the booking type is illustrated. The booking type is divided into two categories, the first is pre-booked, which means that the traffic ride was booked for a specific date and time in the future. The second category is directly booked, which means that the customer requires the taxi presently. It is clear from figure 4.7 that the most popular kind of booking type is directly booked taxi rides.

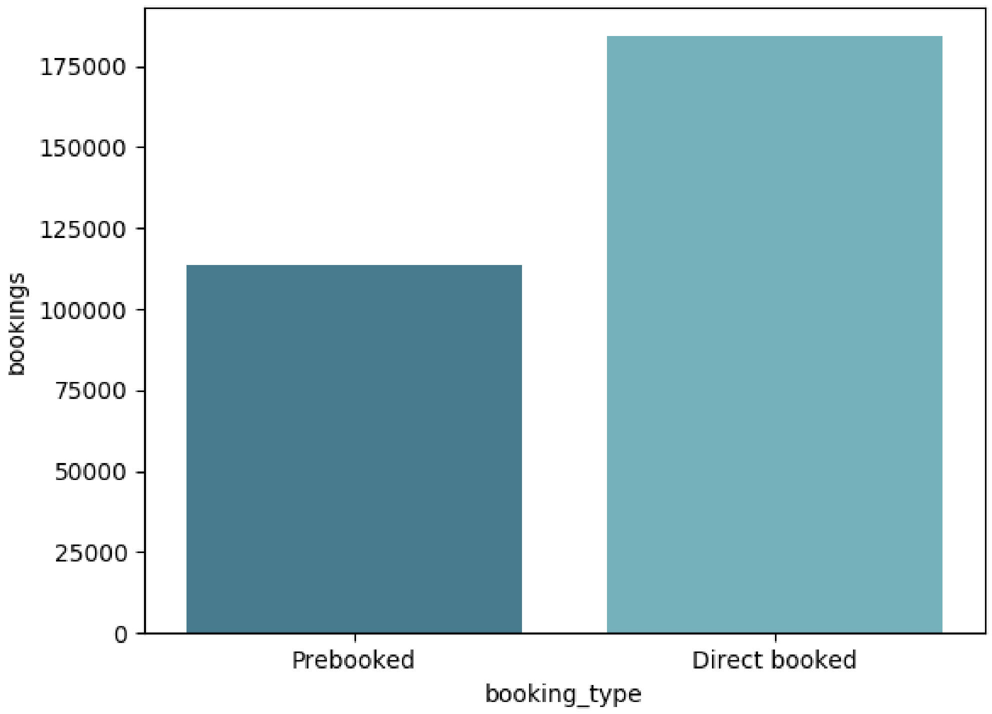


Figure 4.7: Type of booking

In figure 4.8 the distribution of devices used when booking taxi rides are illustrated. There were no initial effects noticed on the overall result when the device aggregation was included in the model. However since there are a numerous amount of rides and it could have effects on individual rides on a lower level, it was kept in the input matrix. What is interesting is to see how dominant the iPhone usage is compared to Android devices, this is approximately the opposite compared to the rest of the world where Android devices dominate the iPhone users.

In figure 4.9 it is possible to see how many users belong to five different categories depending on the number of rides they have completed. The first category contains all the users which have used the Taxi Stockholm service once, and this is the biggest group of users if you divide the users into groups depending on how many rides they have taken. The second category contains all the users that have completed two rides and the third category contains all the users which have completed three rides. There is approximately a halving in the number of users from category one to category two, and from category two to category three. Then the fourth category contains all the users which have traveled between four and ten times and the fifth category contains all the users which has traveled more than ten times.

There was no rigorous performance evaluation of the data aggregation, however there were noticeable differences in the result when they were included into the input matrix. The aggregation of booking type scored approximately a one kilometer improvement in the result from the loss function. The same thing happens when the aggregation regarding time, day of the week, and week of the year is added. Around a one kilometer improvement is noticed.

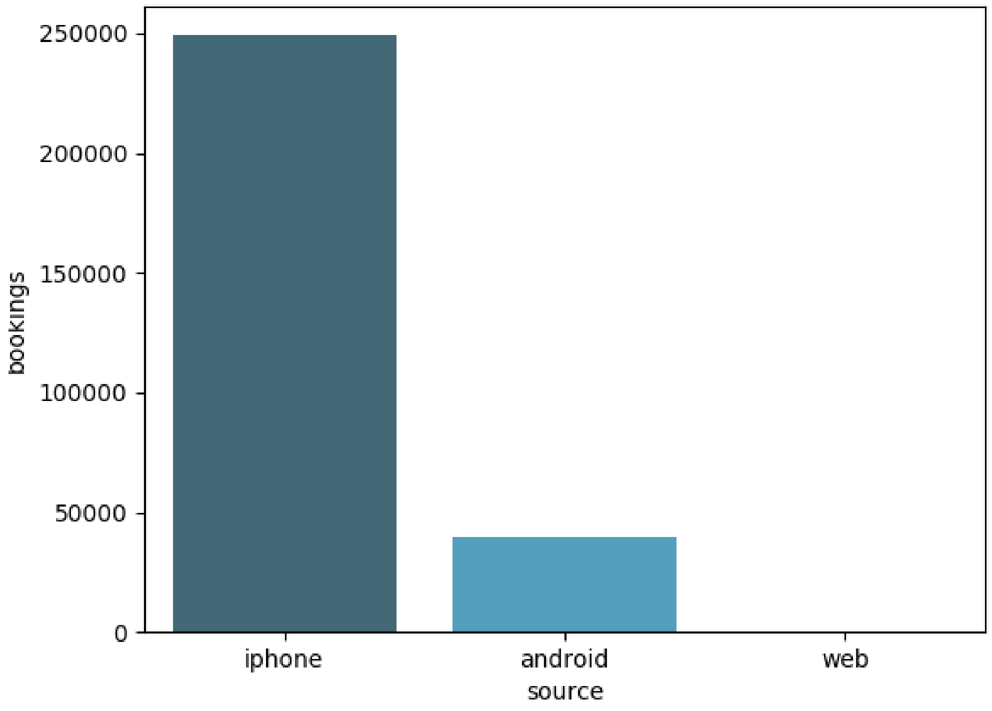


Figure 4.8: Bookings per device

The standardization of the coordinates resulted in an improvement of approximately one kilometer in average distance error. The cleaning of the data gave the biggest improvement, some rides had coordinates from outside of Sweden. Destinations and departures where found in Estonia and Norway which had a negative impact on the total result of the model.

4.2 Data Mining

The results of the grid search process which explored the optimal layer size can be seen in table 4.1 and the grid search process which explored the optimal cluster size can be seen in table 4.2.

In order to determine the optimal size of the model a technique called grid search was used. The final results from the grid search of the model size can be seen in table 4.1 and it is visible that the results do not vary much. The results have been gathered into two different types of results. The column called *loss* contains the final result of the loss function after training the model. The column called *eval* contains the result of the evaluation of the model with validation data that is separated from the training data.

We have divided the models into five different models depending on the size of the layers. Then each model is divided into three different categories depending on the number of layers. There is three different versions of model 1 and the factor that separates them is the number of layers they each have and the size of the layers. In the loss category model 3.2, 4.2, and 3.3 performed the best, however the margin between the top three models and the rest was small. The difference between the best performing model and the worst performing model was approximately five points.

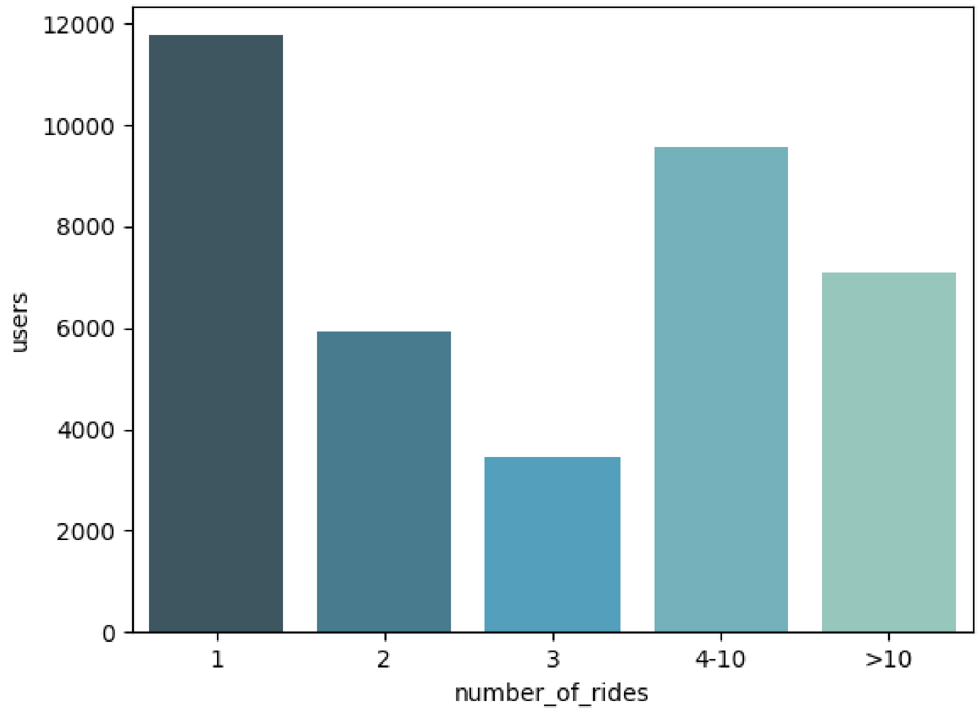


Figure 4.9: User types

In the eval category the difference was even smaller with model 2.3 performing slightly better then the rest of the models. The biggest difference between the best and the worst performing model was approximately two points in the eval category.

The optimal cluster size was determined by a process similar to how the optimal model size was determined. Model 2.3 performed the best on the validation set according to table 4.1 and because if this model 2.3 was used when determining the optimal cluster sizes. The results of the process can be seen in table 4.2 where each model number corresponds to model 2.3 in table 4.1 but with different cluster sizes. The result categories are the same as the ones described in the previous section. There is a new column called cluster size since that is the parameter that was changed during the test instead of changing the number of layers and the size of the layers. It is noticeable that the difference between the chosen cluster sizes does not influence the results of the model to a greater extent. The *loss* column is the model performance using training data and the *eval* column is the model performance using validation data.

The results of the evaluation of the personalized models can be seen in table 4.3. In order to use a high amount of personal data the number of centroids were set as low as 25. From the grid search results in table 4.2 it is noticeable that the number of centroids does not have a huge impact on the result of the model. Since some of the users have a very low amount of taxi rides it would be difficult to generate more then 25 centroids for some users. Even with such a low number of centroids we see two users which were unable to use the models with only personal data.

The user identification column shows the identification number of a Taxi Stockholm customer. The rides column shows how many rides the user has completed. The α and β columns shows the division between personal and general data. The *loss* column is the performance of

Model No.	Layer 1	Layer 2	Layer 3	Loss	Eval
1.1	10	0	0	85.5238	86.4355
1.2	10	5	0	85.6443	86.5514
1.3	10	10	5	85.8144	86.6806
2.1	100	0	0	83.3266	85.6808
2.2	100	50	0	82.6731	85.3008
2.3	100	100	50	81.7630	84.7266
3.1	500	0	0	82.1076	85.6191
3.2	500	250	0	80.7195	85.1762
3.3	500	500	250	80.8425	85.7009
4.1	1000	0	0	81.7478	85.4910
4.2	1000	500	0	80.7339	85.0730
4.3	1000	1000	500	82.8436	86.0855
5.1	2000	0	0	81.9257	86.2821
5.2	2000	1000	0	81.9626	85.7273
5.3	2000	2000	1000	84.0621	86.2256

Table 4.1: Model size grid search results

Model No.	Cluster size	Loss	Eval
1	10	83.7571	86.3780
2	25	81.8572	85.7929
3	50	81.6962	85.8268
4	100	81.5303	85.0416
5	500	82.0409	85.4662
6	1000	81.6932	84.6314
7	2000	81.8901	85.0149

Table 4.2: Cluster size grid search results

the model on training data, the *eval* column is the model performance on validation data. The last row of user 1037395 and user 1042916 has no results and the reason for this is because both users had to few unique rides to create 25 clusters and as such the model did not work for those users.

It is possible to see that most users benefit from some amount of personal data combined with general data. User 1037395 and user 1037340 perform the best with $\alpha = 0.25$ and $\beta = 0.75$. User 1042939 gets the best result on the validation data when $\alpha = 0.5$ and $\beta = 0.5$. Meanwhile user 1089253 perform the best on a fully general model. Lastly user 1042916 performs the best when using $\alpha = 0.75$ and $\beta = 0.25$.

In order to see how well the neural network model for each user performs compared to other data mining methods, they will be evaluated using the test set. Table 4.4 displays the results of a comparison between the neural network model, a random forest model, and a k-nearest neighbor model. The user identification column contains the identification number of a certain user. The *model* column contains the three different models for each user where the neural network is the model that performed best on the validation data as seen in table 4.3. The random forest model uses the scikit-learn random forest algorithm with a maximum depth of ten and the k-nearest neighbor model uses the scikit-learn k-nearest neighbor algorithm with number of neighbors equal to four.

User 1037340 is the only user which benefits from using the neural network model. The random forest model works best for user 1037395, user 1042939, and user 1089253. The k-nearest neighbor model performed the best for user 1042916.

User Id	Rides	α	β	Loss	Eval
1037395	457	1	0	82.6584	67.6501
1037395	457	0.75	0.25	70.5342	49.4677
1037395	457	0.50	0.50	57.8127	44.8110
1037395	457	0.25	0.75	41.7386	44.5210
1037395	457	0	1	-	-
1037340	296	1	0	83.6393	56.3474
1037340	296	0.75	0.25	69.7905	45.4801
1037340	296	0.50	0.50	56.7532	42.0957
1037340	296	0.25	0.75	40.7575	41.9287
1037340	296	0	1	19.8638	194.0030
1042939	202	1	0	82.5538	71.1725
1042939	202	0.75	0.25	71.5122	60.7302
1042939	202	0.50	0.50	58.7562	57.5278
1042939	202	0.25	0.75	42.7973	59.5788
1042939	202	0	1	21.3471	70.6591
1089253	100	1	0	82.9328	53.6089
1089253	100	0.75	0.25	70.3203	55.4688
1089253	100	0.50	0.50	59.2367	59.5948
1089253	100	0.25	0.75	45.7951	63.6724
1089253	100	0	1	34.6354	121.8133
1042916	50	1	0	82.9798	51.1706
1042916	50	0.75	0.25	67.8363	31.3527
1042916	50	0.50	0.50	52.2285	33.9313
1042916	50	0.25	0.75	34.3675	42.1864
1042916	50	0	1	-	-

Table 4.3: Personalization grid search results

User Id	Model	Result
1037395	Neural Network	36.7188
1037395	Random Forest	34.7158
1037395	K-Nearest Neighbor	41.1086
1037340	Neural Network	29.5474
1037340	Random Forest	34.4943
1037340	K-Nearest Neighbor	34.7698
1042939	Neural Network	47.3806
1042939	Random Forest	43.8112
1042939	K-Nearest Neighbor	56.1970
1089253	Neural Network	73.0459
1089253	Random Forest	71.8375
1089253	K-Nearest Neighbor	73.8733
1042916	Neural Network	42.7883
1042916	Random Forest	37.1684
1042916	K-Nearest Neighbor	34.9097

Table 4.4: Model comparison results

5 Discussion

This chapter consists of a discussion regarding what has been done during the course of this thesis. First the results are discussed, followed by a discussion about the methodology. The third section discusses potential future work and the last chapter talks about the work in a greater context.

5.1 Results

A lot of results were gathered during the different phases of the project, some of the early results proved to be useful in the later stages of the project. The initial results were gathered by visualizing the dataset in different ways which proved to be an effective knowledge gathering tool. Before the settling of the final model a tuning phase was conducted which resulted in the performance of several different models. The final results where then gathered with a model which was shaped by the results of the initial data analysis and the fine-tuning of the different parameters.

Data Analysis

The conclusion from looking at figure 4.4 is that the hourly-based distribution of the data seems reasonable. There is plenty of traffic in the morning hours when people is going to go to work. Then the traffic rises during the afternoon as people are leaving work and during the evening there is a major dip which probably can be explained by the fact that people eat dinner around those hours. Then there is a small rise in traffic in the evening again which probably can be explained by people returning home from after work activities.

There was an expectation of increased traffic around three when people return from night clubs which usually close around that time in Sweden. However it should be remembered that these hours are spread out over the whole week. If the same plot had been made but divided into weekends and workdays, then perhaps the traffic pattern would have been different.

By studying figure 4.5 it is possible to find certain patterns in the data that have certain connections to daily life. One explanation for the decreased traffic during Mondays and Fridays could be the close proximity to the weekend. Monday blues is an expression that might not have academic weight behind it but it is discussed among workers and could be one explanation for the low amount of traffic on Mondays. Then during weekends the number of business trips would logically be highly reduced which could explain the drop in taxi rides on Saturdays and Sundays.

Once again there exist explanations for why the graph looks the way it does and the data seems reasonable which is important for the credibility of this thesis. One thing that is strange is that if we compare figure 4.5 with figure 3 in [29] which is a weekday plot of the Porto dataset. It is apparent that they have a smoother graph with less difference between the weekdays and also a higher activity during the weekend. However the smoothness could be explained by the fact that they have a much larger dataset and because of that the difference appears smaller. Also the fact that there is a lower activity of taxi rides in the weekends could be because of cultural differences or that Taxi Stockholm might be more focused on companies as customers rather than private individuals.

Some interesting conclusions can be drawn by looking at figure 4.6. In week 9 there is a big dip in traffic intensity which conveniently can be caused by the fact that most schools in Stockholm had their winter sports holiday during this week. The second big dip occurs in week 15 which also conveniently can be explained by that most schools happened to have their Easter holiday that week. Holidays is often a time when people travel away from Stockholm which would explain the decrease in traffic. During week 21 there is a Swedish holiday called Ascension day which is a shorter holiday then e.g. Easter or winter sports holiday. This could explain the fact that there is a dip in ride frequency but not as significant as during week 15 or week 9.

Then around week 27 we notice another dip which remains until around week 33 which fits in to the period which is usually considered to be a part of the Swedish summer. During the Swedish summer most workers have holidays and children have summer vacation from schools, this seems like a valid reason for the decline in traffic. Another dip is then noticed at week 44 which is perfectly aligned with the autumn holiday that most schools in Stockholm have during this week. Both week 9 and week 48 could be lower then usually due to the fact that the dataset was sliced during some period of those weeks.

After analyzing figure 4.6 the dataset makes sense in a macro perspective and seem to follow the societal trends in Stockholm. It is interesting to compare the distribution found in the Taxi Stockholm dataset with the distribution found in the Porto dataset since similar trends such as holidays or summer vacations are not as obvious in the Porto data. This could be because of cultural differences, or that Taxi Stockholm as a brand is more focused on business trips while the data collected in Porto does not target any specific audience.

The interesting property of the booking type feature is that both categories have rather distinct use-cases. A directly booked taxi ride is more likely spontaneous and less likely to be used in order to arrive at time critical events such as flights, train rides, or important meetings. Meanwhile a pre-booked taxi ride means that you know the time and place beforehand. It seems likely that there would be an increased amount of pre-booked taxi rides that has time critical areas as destinations, such as airports or train centrals. Due to this assumption the booking type was aggregated as a feature into the neural network model.

Tuning Of Hyperparameters

The results of the grid search in regards to model size can be seen in table 4.1. Some interesting phenomenons are present in the results which are usually brought up when talking about the evaluation aspect of neural networks. First of all it is clear that the performance of the loss category is better then the performance of the eval category. What this says is that the model performs better on the training data then on the validation data, which is to be expected to a slight degree. If we look on model 1.1, the difference between the loss result and the eval result is very small, this means that the model does not overfit the training data. Meanwhile if we look at model 3.2, it has the best result on the training data of all the models. However it does not have the best result on the validation data as is visible in the eval column, this means that even though 3.2 is the best model in theory, it is not necessarily the best model in practice due to overfitting.

In the end the model which performed the best on the validation data was the modeled that would be chosen. Model 2.3 in table 4.1 performed the best on the validation data and as such the number of layers were set to three and the size of the different layers were set to 100, 100, and 50, respectively.

At the same time it was a little disappointing to see the small difference between the best model and the worst model. Changes that could have been done is to vary the sizes of the different layers to a greater extent and perhaps try models with more then three layers. However there were hardware limitations from the computer used to train the model. Training the fifth model with three layers was a cumbersome process for the training computer and it is doubtful if it would have been possible to evaluate models of bigger sizes.

Another fact is that one could have spent a decent amount of time experimenting with different layer sizes. But seeing as how small the difference is between the results this did not seem like time well spent. Another fact that supports the result of the tuning process is that in the Porto paper they decided to only use a single hidden layer with 1000 neurons, so their model is smaller in regards to the number of layers, but bigger in regards to the size of the layers. But the difference in model size does not vary greatly which tells us that the size of the model in this study seems reasonable.

From a critical point of view the results does not vary much. The greatest difference is approximately two points between the average error of the best model and the worst model. In some cases the difference is as small as a 0.1 point gain between models. These minor differences in the result could have been caused by random variances in the trained models, and not by the fact that one model is superior to another model.

The results of the grid search of the different cluster sizes can be seen in table 4.2. It is visible that there is no clear pattern and the biggest difference in performance is approximately two points between the best and the worst model. This is a miniscule difference between the models and it is difficult to motivate the choice of one cluster size over the other from a model performance perspective. However it should be said that there is a significant difference in the performance in regards to speed of the model. A model that has a big number of cluster centroids takes longer to execute when compared to a model with a small cluster.

Personalized Neural Network Model

The results for the personalizing of the dataset can be seen in table 4.3. It is possible to see that all the users get a better performance the more personalized the dataset becomes except for the user with 50 rides. User 1037395 and user 1042916 had too few unique rides to have a fully personal model since it was not possible to generate 25 centroids from the available data. User 1089253 was the only user which got a worse result by using personal data.

It is difficult to analyze why that happened, intuitively the reason for the behaviour is because the rides in the test set is very different from the personal rides in the training set. If all the taxi rides in the personal data had the airport Arlanda as a destination, and then all the taxi rides in the test set had Södertälje as destination. Then a model with a lot of personal training data would be more inclined to suggest Arlanda while a general model would be more inclined to suggest a destination in central Stockholm. In that case the model would perform worse the more personal data was used as compared to general data.

User 1042916 and user 1042939 saw an increase in performance however the increase stagnated earlier then the models with more personal data which was expected. The hypothesis from the beginning was that models with many previous taxi rides would benefit more from a heavy emphasis on personal data since after a while patterns will develop. Meanwhile for a new user the hypothesis was that general data should be incorporated in order to avoid a cold start. A cold start is when a user with no previous ride history does not receive any travel suggestions because there is no history of them using the service.

5.2 Method

Numerous insights and thoughts were gathered during the work with this thesis paper. The work flow began with rigorous preprocessing of the data which included conversion of street addresses into coordinates and removal of faulty coordinates that existed in the dataset provided by Taxi Stockholm. Following the preprocessing was a feature engineering phase which involved finding patterns in the data in order to increase the performance of the model. Once the data that was going to be used to train and evaluate the model was finalized, it was time to optimize the size of the model. The hyperparameters of the model were fine-tuned with the help of a technique called grid search.

Data Cleaning

Unfortunately there was a lot of entries in the dataset that were incomplete and had to be removed. In the first wave of cleaning the taxi rides which did not have a user identification, or a destination address, or a departure address, were removed. One potential negative effect of the decision to remove all taxi rides without a user identification would be that there is less taxi rides to train on which still could have been used as part of the general training data. The author feels that the remaining data is sufficient for the experiments of this thesis, however it could be something to keep in mind for future works in the same field.

Once the dataset had gone through an initial cleanse it was time to convert the departure and destination addresses into a format which made them useful. For this task a technique called geocoding was chosen which seems like a reasonable approach. However it turned out to be complicated to geocode such a big set of coordinates without paying any money. The first alternative which came to mind was Google Maps, and it worked well in the beginning and the resulting coordinates from the geocoding process were really good. However, it turned out that you are only allowed to send 2,500 requests per day to the Google Maps geocoding service. At such a rate it would have taken approximately 240 days in order to geocode the whole dataset and that was clearly not an alternative.

Another option was to use an open-source service called Nominatim, there was no daily limit on the usage of Nominatim.[1] They say in their usage policy that bulk geocoding is not advised and refer to third-party services for that kind of functionality. It is also possible to set up a local instance of Nominatim. This would perhaps have been the best option in hindsight, however it felt like a lot of work at the time when the decision was made. There was a desire to not put to much time into the geocoding of the coordinates since it could potentially take a lot of time away from the primary task of the thesis. Another argument which is the strongest argument against using Nominatim was that the resulting coordinates were not as exact as the coordinates from Google Maps or Bing Maps.

The final option, and the option that was used in this thesis was Bing Maps.[2] Bing Maps provided a level of accuracy in its geocoding results which were comparable to Google Maps and it allowed for up to 50,000 requests per day. Instead of taking 240 days it would take 12 days to geocode all the addresses and the resulting coordinates would have a better quality then the coordinates that Nominatim would have delivered. It seemed like a favourable compromise, however it could be of interest for future studies to keep in mind that there exist several different geocoding services available and they all have strengths and weaknesses.

After the conversion of street addresses to coordinates a lot of work was done in pursuance of trying to understand the data better. This was done by plotting the data in different manners and thanks to a scatter plot of the spatial distribution, it became clear that there was still a lot of data that was faulty. Some of the taxi rides had destinations or departures outside of Sweden e.g. location such as Norway and Estonia was found. In order to resolve this issue a simple geofence cleaning was done with the remaining data. The area of the geofence was not chosen according to any scientific standard but was approximated by looking at the spatial distribution of the data. This approach could have been improved by being more exact when choosing a geofence area by using a technique such as the BoundingBox tool which allows you to draw a box over a map and then provides you with the coordinates for that box.[3]

Some of the features were removed during the cleaning of the data. Both the features which involved different Taxi Stockholm zones were removed. There was a lot of rides which

[1] https://nominatim.openstreetmap.org/
[2] https://msdn.microsoft.com/en-us/library/dd877180.aspx
[3] http://boundingbox.klokantech.com/

did not have any zones at all and there seemed to be inconsistencies between the zones. An example is a case where two rides had the same destination address but only one of the rides had a destination zone, this weird inconsistency was the biggest motivating factor behind removing the zones.

In hindsight the zones could have been a good feature to include if they had been consistent over the destinations and departures. One solution could have been to try and map each zone to a unique address and then manually assign each taxi ride with that unique address to the zone which belogs to the unique address. In the Porto paper they used designated pick-up zones and drop-of zones and it had a positive effect on the performance of the model. In the end the work needed in order to complete the zone assignments seemed like a cumbersome task that was not worthwhile so it was skipped, but not overlooked.

The passenger and the phone features were also removed. The motivation behind this decision was mainly that the user identification, the phone, and the passenger features would be the same for each unique user. Minor experimentation was done by trying to simulate how it would have been with the phone and the passenger features included, however no major impact on the results were noticed. It is difficult to imagine a scenario where the features would have a major negative effect, perhaps if a person changes its phone number. Then there will be two very similar but nonetheless slightly different data subjects which might be treated differently by the model, when the travel pattern should not be influenced by a simple change of phone number.

Data Transformation

One important component of the thesis methodology was the aggregation of the input data of the model. The raw dataset content is displayed in table 2.1 and before anything was entered into the model all the features were transformed or removed from the input. The final feature set which was given to the model can be seen in table 3.1.

The user identification feature was originally represented as an integer, and it was always in the scale of one million. If the user identification was zero, then the taxi ride was not bound to a specific user. During the early stages of experimentation with the model there was a problem related to the user identification feature. After some experimentation it was assumed that the size of the user identification feature caused problems with the model. At first there was an attempt at binarize the user identification, this means to convert all the different user integers into a binary representation. Unfortunately there is over 50,000 unique users in the dataset and this proved to be to much for the memory of the laboratory computer to handle.

Instead a function called MinMaxScaler from scikit-learn was used in order to transform the big user identifications into a small number with a range between zero and one. After implementing this data aggregation the previous problems that the model had were resolved. It was difficult to record the performance gains from aggregating the user identification, however it could be seen as a performance boost to transition from a nonworking model into a working model.

First the destination and departure addresses were converted into coordinates with the help of geocoding, which is discussed in greater detail in the previous section. The coordinates were processed once again by the means of standardization by removing the mean and scaling to unit variance. The coordinates used in the Porto paper were also standardized with a zero mean and unit variance.

The time stamp feature was also aggregated and the results of this this is brought up in greater detail in the previous section. The approach to the aggregation of the time stamp was to look at figure 4.4 and divide the hours into different categories which were binarized. The approach chosen in this thesis had little in common with the approach chosen in a similar paper mentioned in the similar works section[9]. In the Porto paper the motivation for the time parameter was divided into quarter hours since this better describes human behaviour.

It is unclear whether their approach would have worked better in this case and after minor experimentation with whole hours and quarter hours, the categorical division approach was chosen.

In hindsight the time stamp aggregation could have been done better. An assumption was made that all the days of the year can be divided into the same set of time zones. The categorization could have been more specific as in weekends have different time zones as compared to weekdays. It is even possible to be so specific as to give individual days different time zones. It should be said that the effect on the performance is highly hypothetical and it could be that data aggregation on such a miniscule scale could have no effect.

Another aggregation of data was the feature called date, which can be seen in table 2.1. The date feature was divided into two smaller features which can be thought of as weekdays and holidays. The weekdays are divided into different categories and the categories were chosen by studying the patterns in figure 4.5. The categories were then binarized in order to have an equal weight between the different categories.

The aggregation of the weekdays in this thesis can be compared with the technique chosen in the Porto paper. There they simply let the weekdays be divided into individual days and then an embedding of the data was done as compared to the binarization. It is difficult to say which method has a higher performance impact, however the approach chosen by this paper was simple to implement and made sense by visualizing distribution of rides over different weekdays. An alternative to the approach chosen in this paper would be to simply binarize the existing days of the weeks and not divide them into different categories depending on ride activity of each day.

The same problems that were brought up in the discussion of the time stamp can be seen here. The aggregation could have been plenty more sophisticated by adding special days such as holidays. There is also other types of data which could have been interesting to add such as election days and other major national events. It is difficult to forecast the effect on the models performance but in future work it could be interesting to include such features.

The second part of the date feature was holidays, unfortunately only a couple of holidays were used. The choice of categories were highly influenced by the data visualization and figure 4.6. The categories represent the biggest holidays in Sweden and also the summer period where nation wide activity goes down. The categories were once again binarized. This approach can also be compared with the approach used in the Porto paper. There they simply represented each week by their week number and then embedded the feature.

It can be considered a little sloppy to only have included the big holidays in the aggregation. The only excuse for this was that too much focus was put on the data which was visualized and zero thoughts were put towards looking at specific holidays or major national events. However the categories that were chosen is backed up by the data that could be seen in the graphs. Weeks that could be aggregated that does not contain any of the big holidays is e.g. week 18 which had a Swedish holiday called Walpurgis Night. During week 21 there is another Swedish holiday called Ascension day which could also be used as a category in the holiday aggregation. There is also week 23 which contained the Swedish national day which is a holiday in Sweden. All of these holidays are usually just one day holidays as compared to Easter and the winter sports holiday, however it is possible to see minor dips in activity in week 18, 21, and 23 compared to other weeks of the year.

The passenger type aggregation was done by looking at the number of times that people have traveled and divide them into different categories based on the frequency of travel. The categories are first time users, second time users, returning users, regular users, and frequent users which can be seen in table 3.1. They did not do anything similar in the Porto paper. The division of the categories were done rather bluntly with some minor experimentation. A more rigorous approach would be to divide the passenger types into many different feature sets and test the model with the different feature sets in order to find the best blend of passenger division.

However one can argue that there is a difference between people who travel more and people who travel less. It can typically be seen that people who travel with a high frequency often travel between two addresses, which would indicate journeys between a workplace and a home. One thing that would have been interesting to look as is the distribution of how many different addresses that people who have only used the service once have traveled to. Would home addresses be the most common destinations for people who seldom use taxi rides as a transportation method or would it be common destinations such as Arlanda airport or the city terminal.

The booking device was also added into the final feature set and was aggregated into a binary representation with three categories. The categories were the iPhone, an Android device, or the website. This was not done in the Porto paper and in retrospect it does not seem like an important feature. It was included and it at least sheds light on the mobile phone market in Sweden which is opposite to the rest of the world. A plot of the distribution can be seen in figure 4.8

The last aggregation was the booking type which is either a pre-booked or a directly booked taxi ride. The distribution can be seen in figure 4.7 and it was not a feature that they used in the Porto paper. The booking type of a taxi ride seems highly relevant and it was a single binary number where a one represents a pre-booked ride and a zero represents a directly booked ride. It is difficult to find a better way of aggregating this feature and considering the discussion in the results section it was relevant to keep it in the final feature set.

Tuning Of Hyperparameters

The tuning of the hyperparameters was an important process that helped determining an approximate optimal size of the model. A technique called grid search was used which means that different sizes were specified and then evaluated in order to find the size which worked the best. The approach was to pick a range of sizes with inspiration taken from previous research within the same subject area, such as the Porto paper. The grid search technique seems like a reasonable approach to the task of finding the optimal model. The ranges are decided with guesswork and intuition which does not appear very scientific at fist, but if the ranges are big enough and if inspiration have been gathered from similar studies, then the approach seems feasible.

In the end the number of centroids were set to 25 because the amount of unique destinations was so low for each user that it would have been impossible to generate 1000 centroids for them. In a way the tuning of cluster size was not really necessary to do, however the results in table 4.2 serve as proof that there is not a huge performance loss between using 1000 or 25 centroids.

The Model

One important aspect of the model is the loss function which is used to minimize the training error of the model. Three different loss functions were discussed in chapter 2 and two of them were evaluated during the construction of the model. Initially the Haversine distance formula which can be seen in (2.5) was implemented since this was the formula that was recommended in the 2015 Kaggle taxi prediction challenge[9]. The Haversine loss function did not perform well together with the model and the data that was used in this thesis. The exact issue of using the Haversine distance remains unclear, the theory is that the result of the equation inside the square root becomes a negative number which can be seen in equation (2.5). The error manifested itself by producing the value *nan* during the training of the model and in the past that error has been produced by having null values in the input values of the loss function.

The Haversine distance formula was replaced by the equirectangular projection formula which is similar to what they did in the Porto paper[9]. This had a positive impact on the training of the model since it was now possible to train the model without producing *nan* values. In equation (2.6) it is possible to see that the value of the equation inside the square root cannot become negative since both the x value and the y values is to the power of two.

Neither the Haversine distance loss function or the equirectangular projection loss function existed in the Keras library. They had to be implemented manually which proved to be a surprisingly cumbersome task initially. Since the Keras library is built upon TensorFlow the loss function had to be implemented using TensorFlow functionality which was not self-evident how to do in the beginning. The object that you do all the computation on is called a tensor, and the problem was that traditional operations found in the Python libraries does not work with tensors. Instead you have to use functions found in the TensorFlow library which resulted in some confusion during the early stages of the project.

Another important component of the model is the optimizer function which was brought up in chapter 2. All three optimizers brought up in chapter 2 were evaluated during the course of this thesis project. Initially stochastic gradient descent with momentum was used as optimizer which worked well and it seemed to perform better than both Adam and RMSprop. Then when the coordinates were standardized something happened and it did not work well at all, the learning rate started to increase instead of decrease. The same thing happened to Adam and meanwhile RMSprop worked well so it was decided to use RMSprop instead of either stochastic gradient descent or Adam. Unfortunately the reason for the strange behaviour of the optimizers was never located.

The hyperparameters used by the optimizer was never fiddled with and the standard values presented by the Keras library were used. The motivation behind not evaluating the performance difference of using different values for the hyperparameters was lack of time. Another reason is that there were other areas within the project which could have been implemented better which probably would have had a higher impact on the result then the hyperparameters of the optimizer.

The first two hidden layers have the rectifier linear unit activation function while the layer before the softmax layer has the activation function tanh. The reason for this is because a strange behaviour was noticed when the data became highly personalized and three rectifier linear units where used before the softmax.

In order to make the model more personal the first obstacle that had to be dealt with was a class imbalance problem. There were 295,940 total taxi rides and the user with the most rides had 457 rides. In order to solve the class imbalance problem the personal travel data had to be increased. However this goal had to be accomplished without increasing the total dataset size since the tuning of the model had been done with a dataset size corresponding to the total number of taxi rides. The solution to the problem was to oversample the personal dataset and to undersample the original dataset and then add them together. The personal data was oversampled by duplicating the existing data and the general data was undersampled by picking random samples from the dataset.

Both the oversampling and undersampling technique have backing in scientific literature, however there exist more sophisticated methods then just duplicating the available data. One technique called SMOTE involves creating copies of the data with minor modifications[15]. In the context of taxi rides one example would be to duplicate taxi rides and change the original timestamp with a couple of minutes so that there is a difference between the original taxi ride and the duplicated taxi ride. With more time this technique could have been worth checking out as a tool to increase the personal dataset.

One thing that should be kept in mind is that the models which were trained with purely personal data is overfitting the data. What this means is that the purely personal model performs the best in theory but perhaps not in practice. However it depends on how often the user travels to new destinations, if the travel pattern does not change then the model will perform well. The solution to the overfitting problem would be to use a dataset which also contains some taxi rides from other users.

The Evaluation

Everything from features, loss functions, optimizers, model sizes, and datasets have been evaluated throughout the work of this thesis. The evaluation of the features was done by running continuous experiments and trying out different settings. After a while no improvement was noticed when features were added or removed so that is when the final feature set was finalized. The evaluation of the hyperparameters was done by running tests of different models and comparing the score which the model received on the validation set.

Once the hyperparameters were set it was time to evaluate the effect of the personalization of the dataset had on the neural network model. Each user had five models that were trained with varying degrees of personal and general data. Then those five models were evaluated using a validation set which consisted purely of personal taxi rides. This was done in order to compare the performance of the five different models on a dataset which would mimic a real life scenario. One of the models performed the best for each user and was chosen as the final neural network model for that specific user.

The neural network model which performed the best on the validation set for each user was then compared to a random forest model and a k-nearest neighbor model. This was done in order to get a perspective of how well the neural network model performs when compared to other machine learning models. The random forest model and the k-nearest neighbor model was chosen because they were fairly simple to implement. The comparison was done by evaluating the three models using a unique test set consisting purely of personal data.

5.3 Future Work

During the process of conducting this study there were a couple of things that would have been interesting to research in greater detail but due to lack of time it was not possible. Hot-spots or points of interests is one such area which would have been interesting to look at during the data analysis phase of the project. It would also have been fascinating to attempt to improve the results of the model by creating a top n prediction using the coordinates outputted by the model. Comparing the models performance when using different datasets and studying the effects of using features that were unique to the New York and Porto dataset would also have been interesting. Finally it would have been interesting to try embedding instead of one hot encoding to transform the features.

Hot-spots

There are some things that could have been improved and some things that were not evaluated during this thesis work due to time limitations. One of those things that would have been really interesting to take a closer look at is so called hot-spots or points of interest. By looking at the most frequently visited destinations in the dataset it would be possible to categorize certain destinations as hot-spots. By looking at figure 4.3 for example, it is possible to tell that Arlanda airport would be one hot-spot, and central parts of Stockholm would contain many points of interest but the train terminal would probably be one of them. Then it would be possible to categorize taxi rides depending on whether they are supposed to go to a certain point of interest or if it is going to a random address.

Once the rides are divided into points of interest, it might be feasible to assume that the remaining rides are either home destinations or office destinations. If temporal data is included into the equation, it would probably be possible to predict which destinations are home addresses and which destinations that are office addresses. Assuming that rides in the morning are usually departing from a home address and that rides in the evening are usually destined to a home address. This is similar to what Uber does today in their destination prediction algorithm as described in a blog post[40].

It would have been interesting to look at individual cases where a user has used the service frequently and look at the destinations and departure addresses. It does not seem far fetched to assume that there would be two addresses that are visited frequently which would be the home and office addresses. Then there would probably be a couple of addresses that match the hot-spots generated from general traveling data. Then it seems feasible to assume that there would be outlier addresses such as random visits to friends and family.

One simple way of implementing this would be to look at the ten most popular destinations and departure locations and call them hot-spots. Then each taxi ride could be labeled as a taxi ride which is departing from a point of interest, or a home address, or an office address. Each point of interest could have its own category and then all the home addresses could be grouped into one category and all office addresses could be grouped into one category.

Top N Predictions

A natural extension to the work done in this thesis would be to generate a top n prediction in order to increase the accuracy by presenting more alternatives to the user. Currently the model outputs one pair of coordinates which can be called super coordinates. However due to the natural difficulty of predicting one exact location with such a large set of different destinations, the super coordinates are seldom so accurate that it predicts the exact location that the user would like to go to. A solution to this problem would be to measure the distance from the super coordinate to the users previously visited locations and other points of interests which were discussed in the previous section. It would then be possible to create a top five destination prediction by picking the locations closest to the super coordinates.

Another technique which could be combined with the distance measure from one super coordinate to previous destinations, is what the engineers at Uber call the donut[40]. The donut technique implies that there is a low probability of the user wanting to take a taxi ride to a destination that is shorter than e.g. 400 feet or further away than e.g. 400 miles. This creates an area around the user that looks like a donut and any previous rides or points of interest within the interval are of interest to the algorithm used to predict the destination. Any points of interest or previous destination that exist outside the donut can be excluded due to the unlikelihood of the user wanting to travel to any of the destinations outside the donut.

Stockholm, Porto, and New York

During the pre-study three different datasets containing taxi ride information was found. The model was built and trained with the Taxi Stockholm dataset in mind and used a feature space that was unique for the Taxi Stockholm dataset. However it would have been interesting to build a general model with feature assets shared between the dataset from Stockholm, Porto, and New York. Then evaluate the result from the general models and see how well such a model would transition between different cities. Uber speaks of the challenges in opening up their business in new cities[40], and if a general model with positive performance could be built, then it would provide a foundation for taxi predictions in a city with no previous travel history.

The Taxi Stockholm dataset had a couple of features which was unique, but the same goes for the Porto dataset and the New York dataset. One such feature is called stand identification which was used in the Porto paper and it had a positive effect on the performance of the model. Another powerful feature of the Porto dataset is that it contains a GPS trace of the whole taxi ride, and it is known that each GPS location is recorded with a 15 second interval. This makes it possible to calculate the length of the taxi ride in time.

The New York dataset contained a pickup time and a drop off time, it also contained the trip distance in miles. Two other features which were unique to the New York dataset was fare amount and passenger count. The passenger count could be interesting since it could be a distinguishing factor between family rides and business rides. The fare amount is in a way another distance measure which means that the length of a New York taxi ride could be measured in time, distance traveled, and cost.

One Hot Encoding Versus Embedding

Data representation was one issue which manifested itself during the work with this thesis. Originally the data was fed to the model in a raw format e.g. time was represented as hours which was an integer between zero and 23. Instead of using the discrete variable as input it was decided to represent some of the input variables using one hot encoding or a 1-of-K scheme as it is called in the literature[2]. One hot encoding was a very comprehensible way of representing the features.

One can think of one hot encoding as a representation of states e.g. weekday has four different states, Monday, Midweek, Friday, and Weekend. If the taxi ride was carried out on a Monday then the state of weekday would be *(1, 0, 0, 0)*. One of the strengths of one hot encoding is that it is straight forward when the number of states are few. Another perk is that it engages all the features, instead of just saying that it is Monday, it also says that it is not Midweek, Friday, or Weekend. The one hot encoding technique was not applied to all the input features but it was used on the features which were easily divided into different categories.

In the Porto paper a technique called embedding was used in order to represent the input data[9]. During the early stages of the thesis there was a small attempt of implementing embedding. However it proved to be cumbersome and since the knowledge about the actual effect of embedding was small, it was decided to skip embedding for the time being. In the end it was not implemented at all and since the Porto paper received a slightly better result from their model it would have been interesting to try embedding and see what effect it would have on the result.

One of the perks of using embedding is that it allowed the Porto paper to have more detailed representations of e.g. time, which they divided into quarter hours which is a number between zero and 96[9]. If the same level of detail would have been used with one hot encoding the result would have been a vector of size 96. Considering that the current input vector is of size 28, adding 90 more values would be a significant increase of the input vector and it is difficult to tell if that would have a positive or a negative effect on the result. If the same process is repeated for weekdays, holidays, and passenger types, then the input vector would become very big.

It is possible that some of the granularity of the data is lost while using one hot encoding. However intuitively it works well for representing patterns in the data and the input makes sense. Meanwhile in the Porto paper the black box of the model is increase by obscuring the input data by using complex data representation. It would have been interesting to study the effects of one hot encoding and embedding in greater detail and perhaps compare the performance between the different methods.

5.4 The Work In A Wider Context

This section discusses how the techniques used throughout this thesis project can be used to impact society at large. A process called data mining was a central part of this thesis and it is important to talk about the ethical dilemmas which is associated with mining personal information. Going from taxi destination predictions to full scale traffic predictions and traffic forecasting does not seem like a giant leap. Examples of this is already being used in Seoul of South Korea to great effect.

Societal Effects Of Traffic Prediction

The societal effects of predicting a taxi destination using old travel data might not be huge. However traffic predictions on a larger scale incorporating live data could be of benefit to society. Seoul, the capital of South Korea, is an example of a city which uses traffic forecasting in order to make travelling more efficient. There is a company called Korea Expressway Corporation which is responsible for maintaining South Korea's toll roads. They started a project in 2007 with the goal of predicting traffic in order to prevent congestion. The predictions are used by suggesting routes which would prevent a congestion. However there is a fine balance between redirecting all traffic to the suggested route and cause a congestion there instead of causing a congestion at the original route.

The technology which enables them to do the forecasts are road tolls which record what type of vehicle and at what time they pass the toll. Real-time traffic news about traffic accidents and road construction is monitored. They use a vehicle detection system which simply consist of traffic cameras at regular intervals which record the speed of a car and the volume of the traffic. Weather data and traffic information dating back to 2007 is also a component of the forecast. Seoul also uses a similar system for their public transportation and statistics show an 4.6 percent increase in the number of buses that arrive in time, a 26 percent increase in average speed of buses, and an 2.6 percent increase in the number of people who rides buses[20].

The improvement in arrival time and increased average speed is evidence of an increase in traffic efficiency. The increase in number of passengers is could be caused by increased user satisfaction which probably can be attributed to better traffic efficiency. The process of gathering real-time traffic information is a process which has become easier and easier thanks to the widespread use of smart phones. Seoul is a practical example of the benefits that can be gained from using available information to manipulate traffic for the better.

In the context of Taxi Stockholm and the model that has been used during this thesis, the result of the destination prediction could be manipulated to favor traffic conditions. Once the prediction is complete it could be compared to real-time traffic information and perhaps suggest a destination that is close enough to the destination that the user would like to arrive at that it is possible to walk to. But further away from the location with high traffic pressure, the incentive for walking the last distance would be a faster arrival time and perhaps a cheaper taxi ride. It would probably be possible to create a light traffic prediction system from the taxi data in the dataset, however it would not take civilian road users or public transportation into consideration.

Ethics And Morals Of Data Mining

One topic that is difficult not to discuss when talking about data mining is the ethics and morals of gathering data about people and their behaviour. A lot can be learned about a person by looking at their digital footprint, or travel history, or message history. Due to these issues it is important for companies to remain transparent in order to not lose customers who value their integrity. All the personal data that have been used in order to conduct this study has been anonymous. Either in the sense that the phone number and the passenger name has been hashed in order to both keep the value of the data, and to keep the user anonymous.

The user identification number is an arbitrary number, so it is possible to say that user 12345 has traveled a lot to these addresses but it is not possible to say who user 12345 is. One could argue that a lot of conclusions can be drawn about user 12345 just by looking at how the user has traveled. It would arguably be possible to look at usual destinations and departures, then correlate that information with the time that the rides were conducted in order to find out whether it is likely a taxi ride home or to the office. Once a potential home address is found it would be easy to figure out who the user actually is.

However it should be said that this type of detailed analysis of individual taxi rides have not been done during the work of this thesis. All data has been handled with care and the only people that have looked at the data in its purest form is the conductor of this study and authorized personnel at both Bontouch and Taxi Stockholm. The integrity of the users is intact but they should also be aware of the data that they provide to companies and the conclusions that can be drawn from it.

25th of May 2018 a new data privacy regulation called General Data Protection Regulation (GDPR) will be enforced by the EU Parliament[28]. The purpose of GDPR is to update the directive which was set in 1995 which currently almost seems like the stone age of the digital era. The idea is to empower data subjects and to give consumers of digital products transparency in regards to what type of information is available about them. GDPR states that data subjects have the right to access the data that a company has stored about them, the data should be provided free of charge and in an electronic format, if requested. GDPR also states that a data subject should have the right to be forgotten, which means that all personal data should be erasable.

Another important fact about GDPR is that it will require companies to notify its users of data breaches which compromises the users data within 72 hours of the breach. This has proven to be an important right considering the recent data breaches which has plagued some of the big companies such as Uber and Equifax. Data portability is another criteria which needs to be fulfilled which means that all data should be exportable to other services if the data subject requests it. Privacy by design is also an interesting concept which will be enforced thanks to GDPR, this will mean that digital service providers will have to build their solutions with data privacy from the ground up.

Equifax is a consumer credit reporting agency which is responsible for providing credit scores and credit reports about its customers. Recently they were subject to a hacker attack which exposed personal information of about 143 million Americans[6]. One of the effects of the data breach is that people with bad intentions can use it for identity theft. The Uber hack was also recently highlighted in media, where 57 million driver and rider accounts were lost to hackers[18].

The hack happened in 2016 but was first announced in 2017 which is very bad since user information has been compromised and users should be notified of this so that they can take the necessary steps to secure their accounts and other accounts with similar passwords as their Uber accounts. According to New York Times, Uber tried to pay the hackers 100,000 dollars in an attempt to make it look like a bug-bounty which is popular among tech giants today. GDPR would have made the Uber cover up illegal.

As data subjects we should not only worry about how the data is used, but also how it is stored. Data that is handled recklessly and systems with bad security could lead to data getting into the hands of people with bad intentions. Another hack in recent years is the Yahoo hack where one billion user accounts were hacked[13]. If we are 7.6 billion people on earth, that is one out of every seven humans accounts that were hacked. Even though it is likely that people have multiple accounts and that a lot of the accounts are inactive, it is still a great deal of information lost to people with bad intentions.

6 Conclusion

A neural network model which predicts the destination which a user wishes to travel to has been constructed. One can argue about whether the results are any good. If the results of this thesis is compared to the results of the taxi destination prediction challenge, then the results of this thesis are not very good. The winning teams final model received a average error of 2.03 which can be compared to the lowest average error noticed in this thesis which was 29.5 for user 1037340.

However it should be kept in mind that there are a lot of factors which play in. For example the city of Porto is much smaller then the city of Stockholm. Also the Porto dataset consisted of over one million taxi rides, while the Taxi Stockholm dataset had approximately 290,000 taxi rides. There is also the fact that there is an error margin in the geocoding process and only the start coordinate was available while in the Porto paper they had the full start trajectory.

The average error works well when comparing different models with each other but since the coordinates are standardized it is difficult to interpret how well the model actually performs in the world. If the coordinates are converted back to its original values and we then calculate the equirectangular distance for each data point in the test set, then we will get the result in kilometers. For user 1037340 the average error is 1.7 kilometers which according to Google Maps is approximately a 20 minute walk. For user 1037395 the average error is 2.1 kilometers which is approximately a 30 minute walk according to Google Maps. A 20 minute walk might not be optimal in an urban environment, but if you consider that the total area included in the data set is approximately 555 kilometers squared it seems decent.

The neural network model has the single best result of all the models. It performs better then the k-nearest neighbor model for four out of five users. The random forest model performs the best overall and beats the neural network model for four out of five users. The neural network model clearly has potential. The author thinks that there still are areas where improvements could have been made, such as using embeddings.

6.1 Research Questions

In the initial chapter of this thesis a series of research questions were formulated. During the course of this thesis attempts have been made to gather answers to the questions that were asked.

- Is it possible to predict a taxi ride destination using the model produced in this thesis and previous taxi ride history?

The neural network model will not be able to predict the exact taxi ride destination that is expected from the test data. However the best neural network model managed to get an average error of 1.7 kilometers which is a manageable distance but still nowhere near a correct prediction. This model could be a good first step in a two model solution. It was discussed in section 5.3 how a top n prediction model could be constructed in order to produce accurate predictions of where the user would like to travel.

- Does personal data improve the destination prediction of the model when compared to the predictions derived from general data?

The answer to this question is in most cases yes, increasing the usage of personal data definitely has a positive effect on the result for four of the five users. The more rides each user has completed in the past the more they benefit from the increased usage of personal data. The exception to this rule is user 1089253 which does not gain anything from using personal data and the possible reason for this behaviour is discussed in section 5.1.

- Is there a threshold for when a model using personal data becomes more efficient than a model using general data?

For user with more than 200 previous taxi rides are benefited when $\alpha = 0.5$ and $\beta = 0.5$. User 1037340 and user 1037395 gain a very small performance increase by having $\alpha = 0.25$ and $\beta = 0.75$, the increase is so small so that they might as well share the same threshold as user 1032939. User 1089253 with 100 previous taxi rides does not gain anything by increasing the amount of personal information. User 1042916 with 50 previous taxi rides achieves a performance boost when $\alpha = 0.75$ and $\beta = 0.25$.

The answer to the questions is that for users with more then 200 previous taxi rides see a performance boost when using a blend of 50 percent general data and 50 percent personal data. For rides with 100 or less taxi rides there is a chance that the user benefits from increasing the amount of personal data, however there is also a risk that a general model will give the best performance.

6.2 Future Work

It would have been interesting to try and make the prediction more accurate by adding a second component at the end of this model which attempts to create a top n prediction. A simple version could be to calculate the distances to previous destinations and pick the five closest destinations. It would also have been interesting to make one model for both the Taxi Stockholm and the Porto dataset and compare the results.

Another interesting future work would be to try and classify taxi rides, for example to try and classify if a taxi ride is a journey home or a business trip. Classifying certain destinations as hot-spots such as airports or train stations through some form of collaborative filtering would also be interesting to look at.

Bibliography

[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016).

[2] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. "Wide & deep learning for recommender systems". In: *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM. 2016, pp. 7–10.

[4] Nitin R Chopde and Nichat M. "Landmark based shortest path detection by using A* and Haversine formula". In: *International Journal of Innovative Research in Computer and Communication Engineering* 1.2 (2013), pp. 298–302.

[5] Sophia Ciocca. *Spotify's discover weekly: How machine learning finds your new music*. 2017. URL: `https://hackernoon.com/spotifys-discover-weekly-how-machine-learning-finds-your-new-music-19a41ab76efe` (visited on 10/25/2017).

[6] Federal Trade Comission. *The equifax data breach*. 2017. URL: `https://www.ftc.gov/equifax-data-breach` (visited on 12/20/2017).

[7] Paul Covington, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations". In: *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM. 2016, pp. 191–198.

[8] Patricia S Crowther and Robert J Cox. "A method for optimal division of data sets for use in neural networks". In: *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer. 2005, pp. 1–7.

[9] Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. "Artificial neural networks applied to taxi destination prediction". In: *arXiv preprint arXiv:1508.00021* (2015).

[10] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "From data mining to knowledge discovery in databases". In: *AI magazine* 17.3 (1996), p. 37.

[11] Scott Fortmann-Roe. *Understanding the bias-variance tradeoff*. 2012. URL: `http://scott.fortmann-roe.com/docs/BiasVariance.html` (visited on 12/12/2017).

[12] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, pp. 315–323.

[13] Vindu Goel and Nicole Perlroth. *Yahoo says 1 billion user accounts were hacked*. 2017. URL: `https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html` (visited on 12/20/2017).

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[15] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[16] Douglas M Hawkins. "The problem of overfitting". In: *Journal of chemical information and computer sciences* 44.1 (2004), pp. 1–12.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[18] Mike Isaac, Katie Benner, and Sheera Frenkel. *Uber hid 2016 breach, paying hackers to delete stolen data*. 2017. URL: `https://www.nytimes.com/2017/11/21/technology/uber-hack.html` (visited on 12/20/2017).

[19] Anil K Jain. "Data clustering: 50 years beyond K-means". In: *Pattern recognition letters* 31.8 (2010), pp. 651–666.

[20] Shin Lee Joon Ho Ko. *TOPIS: Seoul's intelligent traffic system*. 2017. URL: `https://www.seoulsolution.kr/en/node/2595` (visited on 01/21/2018).

[21] Keras. *Keras: Deep learning for Python*. 2017. URL: `https://keras.io/` (visited on 11/13/2017).

[22] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[24] Oded Maimon and Lior Rokach. "Introduction to knowledge discovery and data mining". In: *Data Mining and Knowledge Discovery Handbook*. Springer, 2009, pp. 1–15.

[25] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The Bulletin Of Mathematical Biophysics* 5.4 (1943), pp. 115–133.

[26] Kevin P Murphy. *Machine learning: A probabilistic perspective*. MIT press, 2012.

[27] David L Olson and Dursun Delen. *Advanced data mining techniques*. Springer Science & Business Media, 2008.

[28] European Parliament. *GDPR key changes*. 2017. URL: `https://www.eugdpr.org/key-changes.html` (visited on 12/20/2017).

[29] Julien Phalip. *Kaggle competition report. ECML PKDD 2015 taxi trajectory prediction*. 2017. URL: `https://www.julienphalip.com/blog/kaggle-competition-report-ecml-pkdd-2015-taxi/` (visited on 02/05/2018).

[30] PyTorch. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*. 2017. URL: `http://pytorch.org/` (visited on 11/13/2017).

[31] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor. *Recommender systems handbook*. Springer, 2015.

[32] Frank Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), p. 386.

[33] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ. San Diego La Jolla Inst. for Cognitive Science, 1985.

[34] Daniel Saunders. *The bias-variance tradeoff*. 2017. URL: `https://djsaunde.wordpress.com/2017/07/17/the-bias-variance-tradeoff/` (visited on 12/12/2017).

[35] Umair Shafique and Haseeb Qaiser. "A comparative study of data mining process models (KDD, CRISP-DM and SEMMA)". In: *International Journal of Innovation and Scientific Research* 12.1 (2014), pp. 217–222.

[36] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489.

[37] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. "Mastering the game of go without human knowledge". In: *Nature* 550.7676 (2017), pp. 354–359.

[38] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *Coursera: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[39] Kai Ming Ting. "Precision and recall". In: *Encyclopedia of machine learning*. Springer, 2011, pp. 781–781.

[40] Chintan Turakhia. *Engineering more reliable transportation with machine learning and AI at Uber*. 2017. URL: `https://eng.uber.com/machine-learning/` (visited on 12/13/2017).

[41] Chris Veness. *Calculate distance, bearing and more between latitude/longitude points*. 2018. URL: `https://www.movable-type.co.uk/scripts/latlong.html` (visited on 01/18/2018).

[42] DRGHR Williams and Geoffrey Hinton. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–538.

[43] Rüdiger Wirth and Jochen Hipp. "CRISP-DM: Towards a standard process model for data mining". In: *Proceedings Of The 4th International Conference On The Practical Applications Of Knowledge Discovery And Data Mining*. 2000, pp. 29–39.