CHALMERS

# Neural Network-based road friction estimation using road weather information

Master's thesis in Computer Science and Engineering

FLORIAN MINGES

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2020

MASTER'S THESIS IN COMPUTER SCIENCE AND ENGINEERING

# Neural Network-based road friction estimation using road weather information

FLORIAN MINGES

Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2020

Neural Network-based road friction estimation using road weather information
FLORIAN MINGES



Master's thesis 2020:05
Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Swedish road sign warning of slippery surfaces.

Chalmers Reproservice
Göteborg, Sweden 2020

Neural Network-based road friction estimation using road weather information
Master's thesis in Computer Science and Engineering
FLORIAN MINGES
Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Chalmers University of Technology

## Abstract

The tire/road friction is an important factor for the overall vehicle performance and stability, and is thus an area of interest for both vehicle and tire manufacturers. Furthermore, accurate road friction estimation (RFE) algorithms could become even more important in the future when autonomous driving systems become more common.

The most common limitation of the current RFE models is that they often require very specific testing conditions and tend to struggle when these are not met, which is an issue when trying to generalise the performance for the every-day vehicle usage.

One of the most important factors in tire/road friction is the road surface conditions, which in turn is heavily influenced by the current and past weather. This thesis studies what added value knowledge of the road weather can have on a road friction estimation algorithm, using a simple neural network. We do this by setting up an experiment where we train two models; one that is trained only with data from the in-vehicle sensors, and one that is trained with in-vehicle sensor data and weather data combined.

Ultimately, the added weather data is able to improve the best performance of our model by 3.76 percentage points from 45.27% to 49.03% (8.3% improvement in terms of the number of correctly classified samples) when modelled as a 6-class classification problem. When modelled as a binary classification task ([0 - 0.5) and [0.5+]) our model's performance improves by 21.5 percentage points from 54.47% to 75.97% (39.5% improvement in terms of number of correctly classified samples).

Finally, we conclude that the added road weather data has a positive influence in distinguishing high from low friction values, while struggling with distinguishing the low friction nuances.

Keywords: Road Friction Estimation, RFE, Machine Learning, Sensor Fusion, Road Weather, Neural Network

# Preface

This thesis started during the late summer 2019 and continued until the winter of 2019/2020. Apart from the supervisor at Chalmers, Selpi, it also included the collaboration with two supervisors from the industry: Srikar Muppirisetty from Volvo Cars Corporation (VCC) and Erik Lindbohm from V-Traffic.

# Acknowledgements

I would like to express my sincerest thanks to all my supervisors; Selpi, Srikar Muppirisetty and Erik Lindbohm. Thank you for your trust, for the continued support and for your valuable suggestions throughout the project. Additionally, I would also like to thank Saurabh Gauwande for the insightful discussions about the data used in the project, and helping me to get a good first hands-on experience with it. Finally, I would like to thank my girlfriend and my parents for the continuous support during a tough period of my life.

Florian Minges, Gothenburg, February 2020

# Nomenclature

- CNN - Convolutional Neural Network
- DL - Deep Learning
- LSTM - Long Short-Term Memory
- ML - Machine Learning
- RF - Road Friction
- RFE - Road Friction Estimation
- RMSE - Root Mean Square Error
- RNN - Recurrent Neural Network
- RWIS - Road Weather Information Systems
- SMHI - Swedish Meteorological and Hydrological Institute
- VCC - Volvo Cars Corporation

# Contents

# 1 Introduction

The tire/road friction is an important factor for the overall vehicle performance and stability, and is thus an area of interest for both vehicle and tire manufacturers [1]. A vehicle would be unable to move without the force of friction and only by understanding this force can one aim to wield a precise control of a vehicle's motion. This holds true not only for human drivers, but also for autonomous driving (AD) systems. As such, road friction estimation (RFE) algorithms play an important role as a part of AD systems.

The most common limitation of the current RFE models, is their narrow scope. They often require very specific testing conditions and tend to struggle when these are not met. One such example is the reliance on the tire to have high excitation, like when you brake or accelerate [2]. They are simply not able to estimate the road friction under a constant vehicle speed. Naturally, there have been several studies looking into alleviating this, with varying results [3].

Ghandour et al. (2010) [4] claim that the biggest factor in tire/road friction is the road surface conditions (dry, wet, snow, ice). Knowledge of the road surface conditions can thus be used to estimate the road friction, which is also shown in an example by Roychowdhury et al. (2018) [5]. However, estimating the road surface conditions and thereafter the road friction is not an easy thing to do with only the sensor data that a vehicle can collect by itself. Fortunately, there are other sources of data that are more useful for this specific task.

Weather stations can be found all across the country in Sweden, collecting data such as temperature, humidity, wind and precipitation. Weather stations in Sweden are not only deployed by the Swedish Meteorological Institute (SMHI), but also by the Swedish Transport Administration (Trafikverket). The latter uses its weather stations to monitor the weather conditions on the roads. This can then be used to warn drivers of unexpected weather phenomena, or to better plan the winter maintenance of the Swedish roads. Data from these weather stations are made open and accessible to the general public. With the increased connectivity of today's vehicles, we can thus make use of this information to better predict the road surface conditions and thus the tire/road friction.

Andersson et al. (2010) [6] categorise road friction applications into two main groups depending on their purpose; information applications and vehicle integrated applications. They can each be grouped further into subcategories, as can be seen in figure 1.1.

Information applications are further divided into two groups based on *the recipient* of the information; (1) the driver themselves, or (2) others in the form of fellow drivers or infrastructure. The information to the driver can be either in its raw form, or in a more user-friendly format, for example a visual/auditory/haptic warning when the road friction hints about slippery road conditions. Likewise, the information can also be passed on to a central server infrastructure. From there one could give the same information or warnings to other drivers on the road. It could also be used for example to utilise the collected data to improve road maintenance.

The other major group, vehicle integrated applications, is divided into (1) driver aids and (2) vehicle functions. This division is made based on who controls *the activation* of the functionality. Driver aids, such as for example adaptive cruise control (ACC) or danger ahead cruise control (DACC), are turned on and off by the driver. Vehicle functions, in which the driver does not have the direct control, includes, among other things, stability control functions (SCS) and autonomous collision mitigation (ACM). The performance of such systems can be improved with a more accurate knowledge of the road friction.

## 1.1 Purpose

The main research question of the thesis is:
*What is the* ***added value****, for a road friction estimation algorithm, of fusing road* weather *information from external services with data from in-vehicle sensors?*

The novelty in this approach lies mainly in the use of external information to improve the estimation performance. Until now, the focus in the field has been on utilising only the easily available in-vehicle sensor data. However, that is not always enough to properly assess the road friction, which is heavily influenced by the current and recent weather conditions. Using external weather data is thus a promising step to further help improve the road friction estimation capabilities of vehicles.

If successful, the approach could be applied to improve the RFE performance in the vehicles so that the vehicles can issue more accurate warnings to the driver before and under slippery road conditions. This in

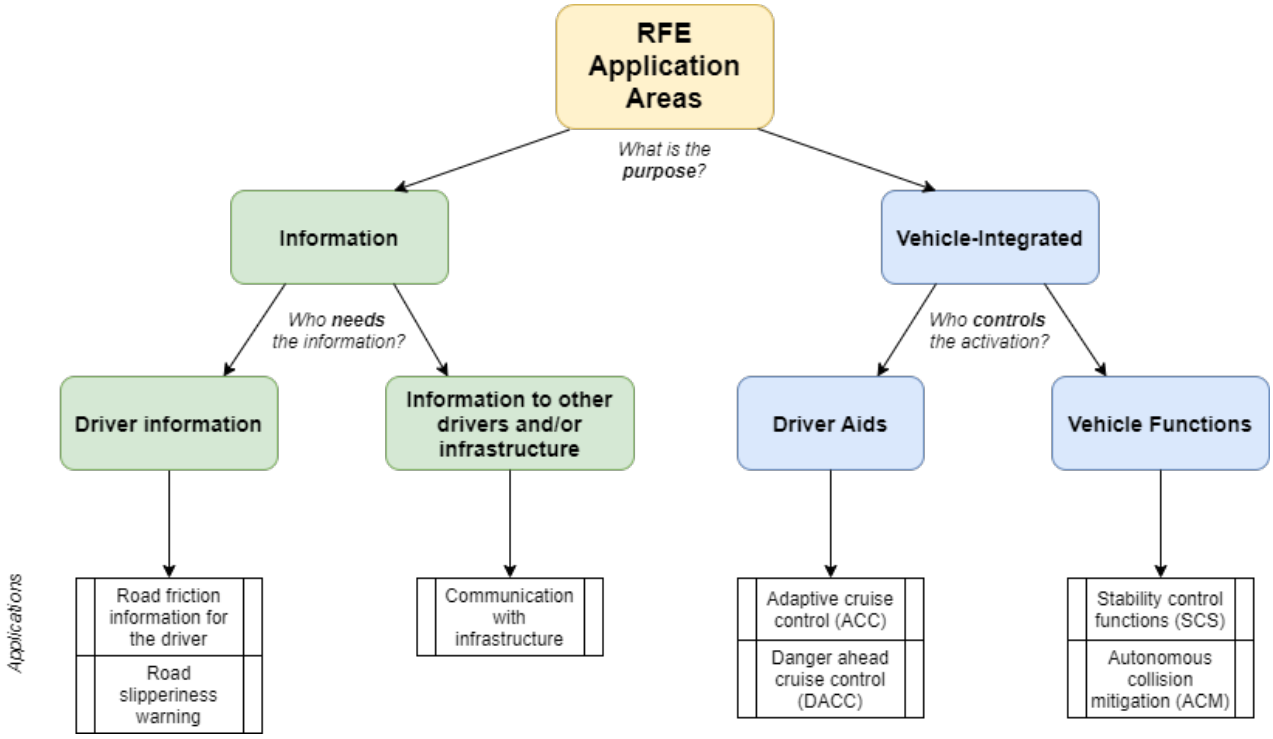



Figure 1.1: Application areas for which knowledge of the road friction can be used. [6]

turn would ideally enhance the driver's preparedness and increase their control of the vehicle, thus helping to prevent crashes.

## 1.2 Scope

The main research question is about the added value that the road weather information can provide. The secondary objective is to have the model perform as good as possible.

The external road weather information that is considered in this project is the data that the *Swedish Meteorological and Hydrological Institute* (SMHI) and *Swedish Transport Administration* (Trafikverket) are collecting and providing through their weather stations. Although other sources of road weather information were considered, they were not used and tried in this thesis. In particular, we initially planned on using a data provider who could give us estimates on the road surface conditions (classified into 11 different classes, eg hard snow, icy etc). This did not go as planned, as their estimations for the location and time of the expedition always estimated the same type of surface condition, which made their data useless in our context. A brief overview of these alternative sources of data can be found in Chapter 2.

The task of estimating the road friction coefficient is treated as a supervised classification problem. It's supervised because the data from Volvo contains the real road friction coefficient values, as measured during their expedition. As a classification problem, the target feature, which is a continuous variable, is also binned.

An interesting aspect of the problem at hand is how big the potential safety benefits of an improved road friction coefficient estimator can be. Improved accuracy does, after all, not mean much if the improved performance does not translate into any specific or concrete added value in terms of reduced fatality rates and/or crashes. Doing this would, at the very least, require cross-referencing with road accident data collected by the Swedish Transport Agency. However, this is deemed outside the scope of this project.

## 1.3 Thesis Outline

Following the introduction, Chapter 2 will elaborate on the current State-of-the-Art in the field of RFE. It is followed by a theory chapter (Chapter 3) on some machine learning principles that are needed to understand

the developed model. This chapter can also be skipped for those that already are comfortable with these principles. Chapter 4 details the setup and methodology used in this thesis. The datasets that were used are also described here. In Chapter 5 we first present an analysis of the weather features and how the weather features were selected, followed by the actual results that were achieved during the project. Finally, this is followed up by some discussion and conclusion in Chapter 6.

# 2 State of the Art

This chapter contains a summary of the state of the art in the domain of road friction estimation. We start by explaining the various categorisation systems that have been proposed to explain the different methodologies. After that we give a brief overview of some of the more recent research in the field. Finally, we talk about the limitations that current methods are facing, and we go over some possible alternative data sources that can be used to enrich vehicle sensor data.

## 2.1 Categorisation Systems

There are many different approaches to doing RFE, and consequently some of the authors surveying the field have attempted to categorise the used methodologies in order to structure them in a logical manner. In this report, we present three different models of doing the categorisation. In a way, they showcase three different points of views one can have when thinking about road friction estimation.

### 2.1.1 Effect- and Cause-based

Müller et al. (2003) [7] suggested to categorise it into *effect-based* and *cause-based* approaches, a division that is also followed by Acosta et al. (2017) [3]. As their names suggest, effect-based approaches measure the *effects* that changes in friction have on various sensor data, while the cause-based approaches attempt to measure things that could *cause* the friction changes, and from there infer the road friction. Each category is then further divided into subcategories based on what is being measured.

Müller et al. (2003) [7] divides the effect-based methodologies into subcategories based on Tire Tread Sensors, Acoustics and Slip. According to Acosta et al. [3], it is precisely these effect-based methods that receive the most development. The cause-based methods, which are often trickier to measure with the vehicle sensors, are divided into subcategories based on roughness and lubricants. An overview of this categorisation system can be seen in Figure 2.1.

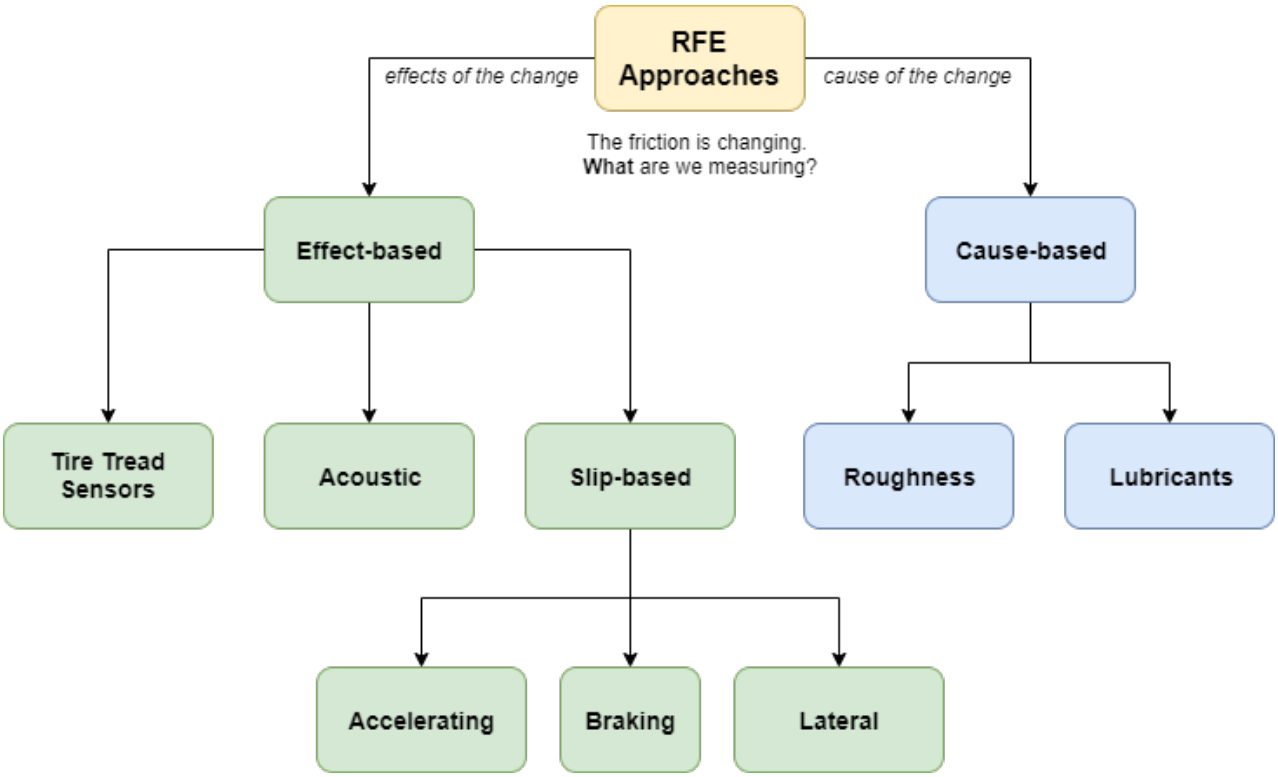



Figure 2.1: Categorisation suggested by Müller et al. (2003) [7] and Acosta et al. (2017) [3]. Figure adapted from [7].

### 2.1.2 Direct and Indirect

A similar categorisation model is presented by Andersson et al. (2010) [6], who categorise the methods into *direct* and *indirect* approaches. Direct approaches aim to utilise vehicle sensor data, such as for example tire forces or slip, to *directly* estimate the road friction. On the other hand, indirect approaches first aim to estimate other features in order to use them for the road friction estimation. An example of this could be using an optical camera system that first tries to identify the road surface condition before using this new information to attempt estimating the road friction [5]. No further subcategorisation is suggested.

The proposed categorisation system by Andersson et al. [6] uses a different split-criteria than the effect- and cause-based division by Müller et al. [7]. Although many of the effect-based methods could be said to be using a direct approach, it would be a gross oversimplification to argue that this always holds true.

### 2.1.3 Model- and Experiment-based

Yet another categorisation model is presented by Khalegian et al. (2017) [1], who divides the methods into *model-based* and *experiment-based* approaches. The former attempts to develop a mathematical model to first estimate certain unknown states, and use these to estimate the road friction. The latter involves making use of the various sensor data collected by the vehicle, such as for example acoustic sensors and temperature sensors, to infer the state of friction-related parameters.

The categorisation system further divides the model- and experiment-based methods into subcategories based on what features they use to make their estimations. This follows the same logic which Müller et al. [7] and Acosta et al. [3] suggested. Model-based methods are divided into (1) slip-slope based methods, (2), tire model-based methods and (3) vehicle dynamic based methods. Meanwhile, the experiment-based methods are divided into ones based on (1) optical sensors & camera, (2) acoustic sensors and (3) tire tread sensors. An overview of this categorisation system can be found in Figure 2.2.

The model-based approach does have a large overlap with the indirect methods, and so does the experiment-based and direct ones. One could argue that there are certain methodologies here that fall in some sort of grey-zone in between, but in general it does match very well.

On a first glance, there also exists some overlaps when comparing this system to the effect- and cause-based division suggested by Müller et al. [7] and Acosta et al. [3].

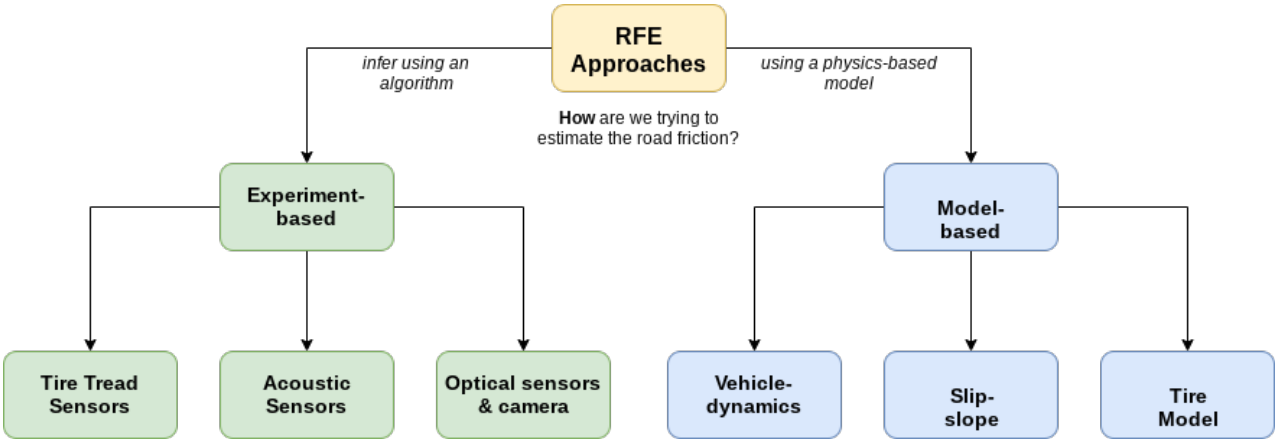



Figure 2.2: Model- and Experiment-based categorisation system.

## 2.2 RFE approaches in Research

The need for more accurate RFE has continuously motivated the research community to improve on its earlier models. However, the underlying mathematical and physical concepts of road friction are still the same today as 20 years ago. As such, we can see that the core ideas of many approaches to RFE in research today do not deviate that much from what was done back then. Improvements have mainly come from a better understanding of the underlying concepts, better algorithms, and improvements in how we measure relevant data. Examples of what specific things *some* researchers have tried to improve in RFE approaches can be seen in Table 2.3. The

table provides a summary of the selected papers' key ideas and performances. The table is a non-exhaustive list, and there are naturally a lot more research papers on this exact topic.

While there are certainly more improvements to be made with these "traditional" models, we can also see a range of research being done today using more "modern" approaches; (1) using data-driven machine learning models and (2) using a connected fleet of vehicles. Some examples of these "new" type of approaches can be found in Table 2.2.

The data-driven Machine Learning (ML) models don't fit very well into any of the system categorisations mentioned in the previous section. Rather they can be seen to partially fit several of the subcategories in the various categorisation systems. Seeing it as a combination of approaches might thus make more sense. While small and very specific ML models can be created to fit into these structures, one of the strengths of ML is that it can make use of a larger variety of data to semi-automatically infer the important features of it. This data-driven approach is thus quite different to the traditional way of doing RFE. That doesn't mean that it's something that is completely new, though.

Data driven machine learning models using the vehicle sensor data have been used for RFE since a long time. One such example is Pasterkamp and Pacejka [14] who used a simple Neural Network for the task as far back as 1997. However, it's only during recent years that these methods have been widely adopted (not only within RFE research), as the computing power and data quantity of that time can't be compared to what we have available today. In certain vehicles we even have access to completely new streams of data, such as for example a video feed of a camera pointing at the road through the front-view window of a vehicle [5], that can be used to better assess the road friction.

Devices and tools across all fields of industry have seen increased connectivity over the last years. This is also being referred to as the "fourth industrial revolution", or just "Industry 4.0". This trend also encompasses vehicles, which nowadays often come equipped with 3G/4G functionality. This opens up the possibility for vehicles to communicate and collaborate with each other for a range of tasks. In a paper by Panahandeh et al. [17], they make use of the connectivity to form a connected fleet of vehicles, which utilises the collected data of a vehicle to warn other drivers of potentially slippery roads. Another paper [13] used the shared data from vehicles to build a high-resolution road condition map, that could similarly warn other drivers of slippery roads once they had been encountered.

The proper use of machine learning models and connected vehicles is something that was not feasible 20 years ago, but it's up and coming and could turn out to be a huge boost for the RFE capabilities in today's vehicles. It also brings a more "modern" touch to the discipline.

## 2.3 Performance

If it was possible to take every result presented by the authors of research papers by face value, it would be fairly trivial to compare them and their performances. However, the fact is that it's almost impossible to compare the results from different papers in a meaningful way. There are a couple of different reasons for that.

Firstly, they almost always use their own custom problem settings. Whether they use a computer simulation, or real data to test their algorithm's/model's performance, it is often done with a setting that is unique to them. Unless tested in the same environments, a comparison thus doesn't make much sense. Furthermore, the research papers which use data collected from real vehicles driving on the road do not typically share these, and the datasets are treated as confidential and proprietary information. As the datasets are also collected independently of others, they are also collected under their own specific set of circumstances, and even the features that are collected might vary depending on the exact source of the data.

Secondly, papers use different target values. The friction values they are estimating are, most of the time, ranging from 0 to 1. However, if they are modelling it as a classification problem, rather than a regression problem, they also need to bin the values into X number of classes. Sometimes it's split into low/high friction, sometimes low/medium/high, and sometimes they use the same classes, but with a different range for the classes. Comparing results from these cases is thus quite hard as well.

Ultimately, the only feasible way of comparing the performance of models/algorithms developed in the various research papers is to actually test them manually on the same dataset, using the same target values. In practice, this turns out to be not that feasible either. The end result is that it's hard to accurately decide which methods are actually the best unless we try them all ourselves.

Figure 2.3: Summaries of some RFE research papers.

| Author(s) | Paper Title | Year | Target Values | Accuracy | Paper Summary |
|---|---|---|---|---|---|
| Gustafsson, Fredrik [8] | Slip-based Tire-Road Friction Estimation | 1997 | 0-1 | Not reported. | Estimates the RF solely based on the wheel slip during normal driving. It applies a Kalman filter and a change detection algorithm to estimate the slip slope, which it uses to infer the approximate road friction. Verifies the correctness with field tests, but uses no proper numerical performance metric. |
| Ahn [9] | Robust Estimation of Road Friction Coefficient for Vehicle Active Safety Systems | 2011 | 0-1 | 0.17 - 0.25 RMS | Combines two methods - a lateral and a longitudinal dynamics based method - to create a more robust RF estimator that covers a wider range of states. They verify the performance with tests on concrete-, snow- and ice- surfaces, both with simulations and real physical tests. |
| Rajamani et al. [10] | Algorithms for Real-Time Estimation of Individual Wheel Tire-Road Friction Coefficients | 2012 | 0-1 | Not reported. | Tests 3 algorithms that each uses a different set of sensor data; (1) using engine and brake torque with GPS data, (2) using torque measurements and acceloremeter data and (3) using GPS and accelerometer data. The algorithms consist of various mathematical models, describing the relations between the sensor data and the road friction. The authors estimate the road friction of each wheel individually with satisfying performance, but offer no numerical performance metrics. |
| Lex [11] | Maximum Tire-Road Friction Coefficient Estimation | 2015 | 0-1 | Not reported. | Models the task as a state estimation problem and applies a particle filter, which is good for noisy or partial data. Does not present any numerical performance metrics, but claim to improve existing techniques. |
| Zhao et al. [12] | Estimation of Road Friction Coefficient in Different Road Conditions Based on Vehicle Braking Dynamics | 2017 | 0-1 | Not reported. | Uses a braking dynamic model of a two-wheeled vehicle to estimate the RF, putting special emphasis on the load transfer between the front and rear axes. Verifies the model with simulations in ADAMS/Car, but provides no proper numerical performance metric. |
| Magnusson et al. [13] | Real-time high-resolution road condition map for the EU | 2018 | 0-1 | 91% of estimations had an absolute error within $\pm 0.15$ when using data from 1 vehicle. 99.7% for 3 vehicles. | Successfully builds a fine-grained friction map over a large area by combining friction measurements from a fleet of vehicles with local road weather models. |

Figure 2.4: Summaries of some RFE research papers that use a data driven approach or a connected fleet of vehicles to accomplish their goal.

| Author(s) | Paper Title | Year | Target Values | Accuracy | Paper Summary |
|---|---|---|---|---|---|
| Pasterkamp and Pacejka [14] | Application of Neural Networks in the Estimation of Tire/Road Friction Using the Tire as Sensor | 1997 | 0-1 | Not reported. | Tests if a NN can estimate the road friction. It can. |
| Pasterkamp and Pacejka [15] | Optimal Design of Neural Networks for Estimation of Tyre/Road Friction | 1998 | 0-1 | Not reported. | Applies a genetic algorithm to find the best possible parameters for a feedforward NN. |
| Song et al. [16] | Road friction coefficient estimation based on back-propagation neural network | 2017 | 0-1 | 0.001 MSE (Simulation) | They used a neural network with 2 hidden layers (each consisting of 10 neurons) which takes 12 vehicle states as input and outputs the road friction coefficient. |
| Panahandeh et al. [17] | Road friction estimation for connected vehicles using supervised machine learning | 2017 | Low/High | 0.19 - 0.37 Error Rate, 0.21 - 0.35 Error Rate 30 min into the future | Using historical friction data from a fleet of connected vehicles and data from weather stations as input parameters, they train an ANN to make binary road friction estimations (slippery vs non-slippery) in the present time, as well as the future (120 min). Due to limited data, they only achieve an error rate of 20-30%. |
| Chen, Shuangshuang [18] | Road Friction Estimation using Machine Learning | 2018 | 0-1 | 0.042 MAE, 0.053 RMSE | Trains an Echo State Network (ESN), a type of low-memory RNN, to predict RF. |
| Song et al. [19] | Estimating the Maximum Road Friction Coefficient with Uncertainty Using Deep Learning | 2018 | 0-1 | 0.057 RMSE normally, and 0.01 RMSE when only using the estimate when certain (88% of all test cases) | They use a neural network consisting of a CNN layer with skip connection, an LSTM layer and finally a fully connected layer, to estimate the road friction coefficient. Afterwards, a Deep Ensemble algorithm is used to estimate the confidence/uncertainty in the predictions. |
| Arvi Jonnarth [20] | Camera-Based Friction Estimation with Deep Convolutional Neural Networks | 2018 | 0-1 | 90% | Uses a CNN to classify images taken through the front window of a car into medium and high friction. Also attempts at enlarging the dataset (which they claim to be one of the main limiting factors) with pictures taken from a racing game. However, this data proves not to be useful. |
| Roychowdhury et al. [5] | Machine Learning Models for Road Surface and Friction Estimation using Front-Camera Images | 2019 | Low/Medium/High | 89% | They apply a 2-step process, in which they first use a CNN to identify the road surface type (4 classes; dry, wet/water, slush and snow/ice) based on front-view camera images, and then apply a rule-based classifier to estimate the road friction (3 classes). |

## 2.4 Limitations

Neither of the approaches are perfect, and they all tend to struggle when the testing conditions change. The most common limitation of the proposed RFE methodologies mentioned in the literature, is their narrow scope. They often require very specific testing conditions and tend to struggle when these are not met. One such example is the reliance on the tire having high excitation, like when you brake or accelerate [3]. Many methods are simply not able to estimate the road friction under a constant vehicle speed. Naturally, there have been several studies looking into alleviating this, with varying results [2].

RFE, and other vehicle safety functions, have always needed quality data to verify their accuracy, but the data-driven approaches of "modern" RFE solutions have also put an increasing requirement on the quantity *and* coverage of the data. In fact, the lack of data can be seen in a lot of the RFE research that has been done, as many of them barely have any real data to work with. Instead they often have to rely on simulation engines to verify their research (e.g. [16]), which can be a bit problematic since those simulations are human-created approximations of the real world.

How much data an ML model needs is dependent on many different factors, such as for example the complexity and dimensionality of the data. However, while not all Machine Learning approaches need huge datasets to perform well, there is a subset of these approaches, called Deep Learning, which seemingly can never get enough data [21]. This can also be seen in Figure 2.5, by [22]. Since collecting data has a cost, one will usually not collect data indefinitely. However, even if the quantity of the data initially seems to be enough, the limiting factor will more likely be the coverage of the data [20]. A machine learning model will usually only learn to predict what it has seen before, and a lack of coverage thus creates blind spots in the model's predictive capabilities.

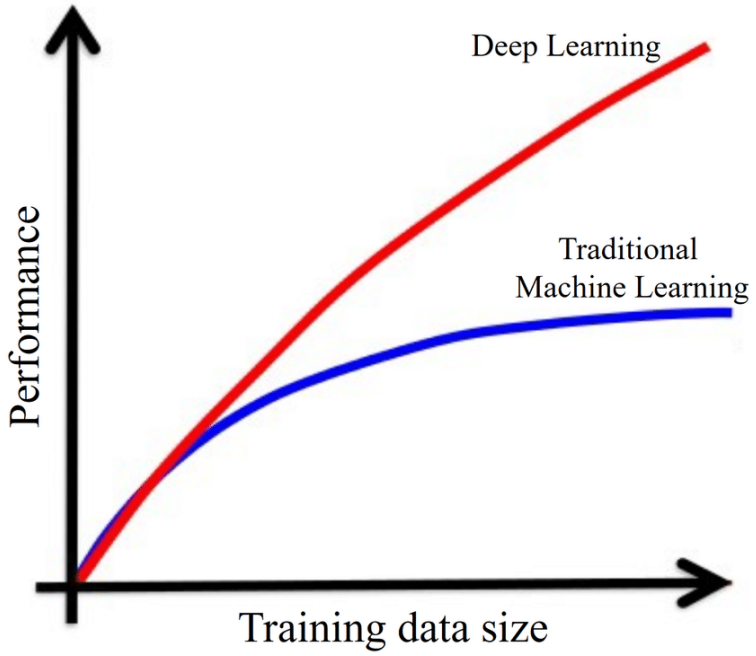



Figure 2.5: Deep Learning model performance based on data size, compared to traditional Machine Learning approaches. Reproduced with permission.[1]

Vehicles today are equipped with sensors that collect a lot of data. However, they are by default not equipped to collect the high-quality road friction data that we want for this research. To do that, the vehicles need to be equipped with certain extra fixings, e.g. a trailer. The more accurate measurement devices are also very expensive and automatic collection of big amounts of friction data from consumers is thus not possible. Instead, one can organise expeditions to collect the road friction data that one needs, which is what the Volvo Cars Corporation did in northern Sweden in March, 2019. The data that was gathered there was then used in

[1]Reprinted from Expert Systems with Applications, Volume 141, Alyafeai, Z. and Ghouti, L., 2020. *A fully-automated deep learning pipeline for cervical cancer classification*, Copyright 2020, with permission from Elsevier.

this thesis.

## 2.5 Alternative Data Sources

It is obvious that researchers need to get their hands on some vehicle sensor data to achieve their goals, but it often stops there. However, there are also other sources of data from third parties that could possibly enrich the information that the vehicle sensor data provides. Many countries utilise so called Road Weather Information Systems (RWIS), that collect various types of information about the roads and the environmental conditions surrounding them. The Swedish Transport Administration (Trafikverket) is for example the institute who is in charge of that in Sweden. Another source of data are Meteorological Services that can provide auxiliary information in the form of general weather status and weather forecasts. One such provider in Sweden is the Swedish Meteorological and Hydrological Institute (SMHI). Yet another possible data source, are third parties that further process the data provided by the aforementioned sources. One such provider is a company called V-Traffic, who based on the information provided by other actors and services, creates an estimate for the current road conditions (e.g. snowy or icy roads).

Another, different data source that can be utilised are other cars. As has been mentioned earlier in this section, there are already research papers that have utilised this in order to get a better grasp of the current road conditions (e.g. [13]), and whether it's slippery or not.

Using these alternative data sources puts some additional requirements on a practical prototype of a solution utilising them, since the data is not originally accessible from within the vehicle. For the prototype to access the information, it might for example require 4G connectivity. This is something that was not easily available 10 and 20 years ago, but which is far more reliable using today's infrastructure.

# 3 Theory

This chapter goes through some of the theoretical backgrounds that can be useful to understand the technical side of the method that was used and the model that was developed in this thesis. Readers who are knowledgeable about machine learning, in particular neural networks and machine learning in general, may skip this section.

## 3.1 Modelling a Machine Learning Problem

Machine Learning is largely a data-driven process. Rather than having the user figure things out, we want to let the data teach the machine how to do things. However, there are some things we need to take into consideration when we are doing this. It starts with the framing of our problem, and continues with how we prepare the data so that the machine can make sense of it. This section will cover some of these steps.

### 3.1.1 Classification vs Regression Problems

Classification problems, in the context of machine learning, is when we try to predict the target *class* that a certain input data belongs to. In these cases we have a predetermined and finite set of classes and the goal of our ML model is to predict the correct *class*. Our target feature here is of the categorical type. In regression problems, we try to predict the exact *numerical value* of the target. We don't have a predetermined range of values, so the target feature is a continuous value.

When the target feature is continuous it might feel natural to model the task as a regression problem. However, regression problems are usually seen as a harder task than classification. To make it easier we can thus, if it makes sense and can still help us answer the question we want answered with our model, bin the continuous target feature into classes. Our simplified model can then be used to classify the target feature into a smaller range of numerical values.

### 3.1.2 Supervised vs Unsupervised Learning

There are different ways how a machine learning algorithm can learn, but the most common ones are supervised and unsupervised learning.

In *supervised* learning we train our algorithm with *labeled* data. That means that each data sample is tagged with what it should be classified as, and that our algorithm can use this information to improve. This is the most common type of machine learning, and the type that is applicable in this thesis.

In *unsupervised* learning we train our algorithm with *unlabeled* data. This means that our algorithm does not know what is "right" or "wrong", and has to figure this out all by itself. This type of learning can be quite complex, but is often used for pattern recognition problems or when big amounts of labeled data is hard to acquire.

### 3.1.3 Data

An ML model can only become as good as the data we input to it. The more data we have, the better, but simply having a lot of data is not enough for an ML model to excel at its task. We need both a sufficient quantity and quality of the data. We not only need our data to be accurate, but also that it should cover as many of the possible real-world scenarios that our model might encounter in the future. Although the goal of an ML model is to learn the underlying representation of the data so that it can generalise this to previously unseen data, this is not an easy task. If the data it has to deal with is very different from the one it was trained with, the model will have problems to accurately do the predictions. It is thus of very high interest to have a dataset with good data coverage.

### 3.1.4 Formatting Data

For a Machine Learning model to make sense of the data, it first needs to convert every data feature into a format that it can handle - numbers. Not all types of numbers are equally useful, and it is also dependent on the context and range of each feature. Most sensor data that we handle is already in numerical format. Non-numerical data is usually categorical, i.e. it has a finite set of values, and we need to transform it to

make it easier to handle. One way of transforming the categorical data is to replace each possible value with a number. However, categorical data does usually not have an ordinal relation between all of its possible values, so we would introduce a misleading bias into the feature by doing so. In ML, what we usually want to do instead is to one-hot encode the feature.

One-hot encoding a feature means that every possible value of that feature becomes a feature in itself, which can only take on two different values; 0 or 1, i.e. not present or present. Thus, a feature X that has four possible values would turn into four binary features, each taking on the value 0 or 1.

#### Scaling Data

Machine Learning algorithms sometimes have problems dealing with data whose feature ranges operate on different orders of magnitude. To combat this we need to normalise the values to a common range of values. This not only helps the performance of the algorithms, but also helps speed up the time it takes to train our network [23]. There are multiple ways how to do this. Some of the common ones are min-max normalisation (Equation 3.1) and Z-score normalisation/standardisation (Equation 3.2).

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3.1}$$

$$x' = \frac{x - \overline{x}}{\sigma} \tag{3.2}$$

In the previously mentioned equations (Equation 3.1 and Equation 3.2), $x$ is the feature value, $x'$ is the new, scaled value, $\overline{x}$ is the average and $\sigma$ is the standard deviation.

In min-max normalisation (Equation 3.1) we scale the values to the range [0, 1] (or a range of our choice, if we can modify the equation slightly). This keeps the relative distance between values intact, i.e. the scale is unchanged.

In Z-score standardisation (Equation 3.2), we change the *mean* to 0 and the standard deviation to 1. This can change the relative distance between values, but is better at handling outliers.

### 3.1.5 Performance Metrics

In order to properly evaluate how well an ML model performs compared to others, we must put a numeric value on its performance based on a common, standardised metric. There are many different metrics one can use, and choosing the appropriate metric is not always an easy task. Each metric has its own strengths and weaknesses at highlighting different aspects of the performance of a model, and the choice of metric is thus heavily influenced by what aspect one finds to be the most important. This might also be influenced by the dataset that is used to train and evaluate the model; an imbalanced dataset, where some types of classes or values are over-represented, might produce drastically different results based on some metrics, compared to a well-balanced dataset.

Depending on if the task at hand is a classification or regression problem, we have to use different metrics to measure the performance of our model. Since this thesis treats the task as a classification problem, an explanation of the different ways of evaluating these can be found below. Regression performance metrics are thus not covered in this report. The metrics are also slightly different depending on if the task is a binary classification problem, or a multi-class classification problem. Both of these are detailed below.

#### For Binary Classification

Perhaps the easiest way to quickly get a visual overview of the performance of a classification model, is to create a so-called confusion matrix. A confusion matrix is a type of contingency table, where each variable is a combination of the class the model predicted for a data sample, and the data sample's actual class. An example of how this looks like in the case of a binary classification task, can be seen in Table 3.1.

Based on this visualisation, it's then easier to explain the various evaluation metrics used for classification tasks. Some of the most commonly used metrics for classification problems are accuracy, recall, precision and F1-score. Assuming that one is dealing with a binary classification task, they can formally be described as:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.3}$$

Table 3.1: Binary Confusion Matrix.

| | | True Class | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted Class | Positive | True Positive (TP) | False Positive (FP) |
| | Negative | False Negative (FN) | True Negative (TN) |

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.5}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{3.6}$$

**For Multi-Class Classification**

For multi-class classification we can similarly build a confusion matrix as before in Table 3.1 for the binary classification. However, here it is a bit harder to precisely define what True/False Positives/Negatives are, as these would change depending on the class one is considering to be *positive*.

Accuracy is the perhaps most straightforward metric for multi-class classification. One simply has to divide the number of correct predictions with the total number of predictions made. On the other hand, precision, recall and F1-score become slightly more complex to compute.

To compute the precision, recall and F1-score for a multi-class classification problem, we start by computing the respective metrics for each class, one at a time, using the same equations as before (see Equations 3.4, 3.5 and 3.6). An example of what can be considered True/False Positives/Negatives in the case of viewing class 2 as the "positive" class, can be seen in Table 3.2. Once we have computed the metrics for each individual class, we have to aggregate them somehow to get an overall metric score. This is usually done in one of two different ways: by either taking an unweighted or weighted average.

Table 3.2: Simplified Confusion Matrix when using multiple classes, as seen from the perspective when class 2 is the "positive" class. Fields that are irrelevant to the precision, recall and F1-score of class 2 are left out of the matrix.

| | | True Class | | | |
|---|---|---|---|---|---|
| | | Class 1 | Class 2 | Class 3 | Class 4 |
| Predicted Class | Class 1 | True Negative (TN) | False Negative (FN) | True Negative (TN) | True Negative (TN) |
| | Class 2 | False Positive (FP) | True Positive (TP) | False Positive (FP) | False Positive (FP) |
| | Class 3 | True Negative (TN) | False Negative (FN) | True Negative (TN) | True Negative (TN) |
| | Class 4 | True Negative (TN) | False Negative (FN) | True Negative (TN) | True Negative (TN) |

An unweighted (macro) average is very straightforward to compute. We simply aggregate the metrics for the individual classes and divide it by the number of classes (e.g. $\frac{\sum \text{precision of every class}}{\#\text{ of classes}}$). However, in cases where one deals with an imbalanced dataset (i.e. where some classes are a lot more frequent than others), it might make more sense to look at a *weighted* average. Here we multiply each metric with the *support* (i.e. the total number of occurrences of a class in the test set), aggregate it, and divide it by the total number of samples in the dataset (e.g. $\frac{\sum \text{precision of every class} * \text{support}}{\sum \text{support}}$).

## 3.2 Neural Networks (NN)

A Neural Network (NN) is a machine learning algorithm that is inspired by how the human brain works and learns new information. It consists of a set of interconnected nodes, so-called neurons, with the ability to pass on information, or a signal, between each other. When a neuron receives a signal from another neuron, it will process the content and (potentially) pass it on to some of the other neurons that it is connected to. While

doing so, it may also change the signal. From a simplified point of view, learning happens when the network of neurons starts to learn how to correlate the input signals with the respective output signals.

### 3.2.1 Basic building blocks

The neuron is the core unit of a NN (see Figure 3.1). It receives a number of input signals, signified by edges in the figure, and transforms these signals before passing on a signal of its own to other connected neurons. Each input signal has a *weight* attached to it, which influences what the neuron will pass on. In ML, these signals are usually all numbers, and the neuron transforms these numbers with a non-linear function of its choice, a so-called activation function (a(y)). Some common activation functions are sigmoid, rectified linear units and tanh. A single neuron learns by altering the weights attached to its input signals.

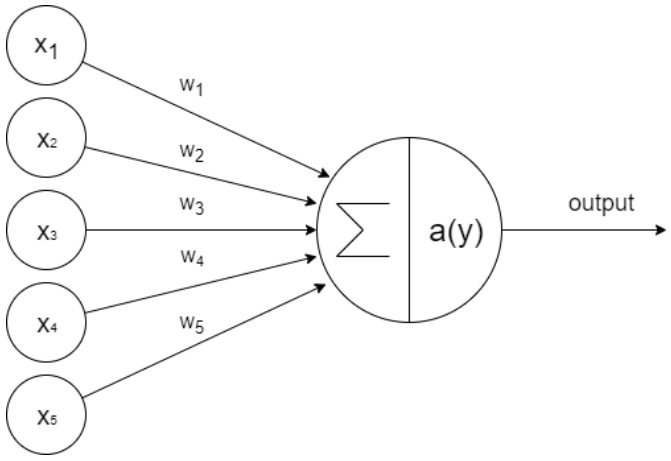



Figure 3.1: A neuron in a neural network.

The most common and simple way of modelling a NN, is to model it as a feed-forward Neural Network. This is a special type of network in which the signals only move in one direction, forward, so that the nodes do not contain a cycle. This lets us model the network in *layers* (see Figure 3.2).

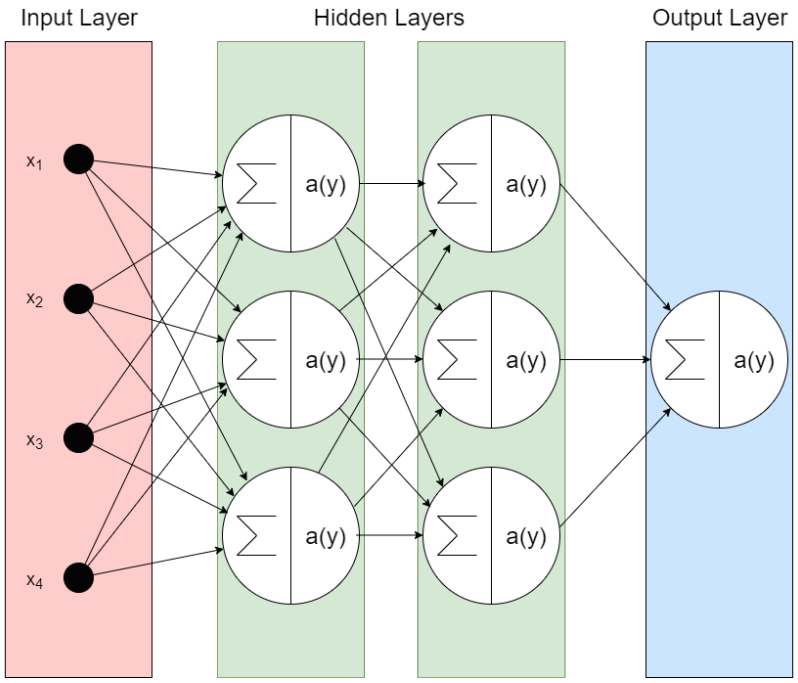



Figure 3.2: Example of layers in a neural network.

Each feed-forward NN consists of at least one input layer, and one output layer. In between these there can be a variable amount of *hidden* layers. Per definition, a NN with at least two hidden layers is also known as a deep NN. This is not a very deep NN in practice, and there exist models that use dozens, or even hundreds, of hidden layers. These type of networks are also known as multi-layer perceptrons (MLP).

### 3.2.2 How a NN learns

The core principle for how a NN learns is by trying to minimise a loss function. When training our network, we pass a data sample through our network and receive an output in our output layer. We then compare this output to what we wanted our network to produce to find our model's error (also known as the loss). By stepping backwards through our network from here, we can calculate the derivative of the error (the gradient) in respect to every weight in our network. This is done with a method called backpropagation, which is an effective system for how to calculate gradients in directed graphs. By doing so, we can adjust the weights by a small margin multiplied by a *learning rate*, to reduce the error/loss for next time. Doing this once for every data sample is called an *epoch*.

When training a NN, we want to keep training until a satisfactory performance has been achieved. Because of that we do not usually stop training after 1 epoch has ended, but rather when the performance of our model is going down. After every epoch we test our model on a validation set, and if we see that the performance is going down compared to previous epochs, we might want to stop our training.

### 3.2.3 Hyperparameters

A NN has many parameters that will influence how well it performs. It is important to differentiate between normal parameters and *hyperparameters*. Hyperparameters are settings that we adjust before we train our model, such as for example the number of neurons and layers our model will have, while the "normal" parameters are things like our model's weight, which will be tuned during training. What we can do to help our model improve is thus how to tune the hyperparameters.

#### Number of Layers & Neurons

The most obvious thing we can tune is directly related to the architecture of our NN, and more specifically, our hidden layers. The number of neurons in the input layer is fixed to the number of features of our input data, and the number of neurons in the output layer is also fixed, depending on the nature of our model. However, we can adjust the number of hidden layers and neurons in them freely. Ideally, we want enough hidden layers and neurons so that our network can learn the underlying representation of the data, but not too many so that we lose our ability to generalise our performance to other datasets.

#### Learning Rate

The learning rate adjusts by how much we will update our weights each time. A larger learning rate thus means bigger changes to the model at each step. The learning rate should be neither too big, nor too small. Having a learning rate that is too large will need less training epochs, but it might also cause the network to find a suboptimal local minima very fast. On the other hand, a learning rate that is too small will take a long time to converge and will need more training epochs, but it might also get stuck in local optimas where it can't get out.

#### Activation Function

As mentioned before, there exist multiple activation functions such as for example sigmoid, rectified linear units (ReLU) and leaky ReLU. They are defined as follows:

$$\text{sigmoid} \longrightarrow f(x) = \sigma(x) = \frac{1}{1+e^{-x}} \tag{3.7}$$

$$\text{ReLU} \longrightarrow f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} \tag{3.8}$$

$$\text{Leaky ReLU} \longrightarrow f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \tag{3.9}$$

Another activation function that is of interest in this thesis is the softmax activation function (see Equation 3.10). It is often used as the activation function in the output layer when dealing with a classification problem. It essentially takes the output values of the output layer and transforms them into a probability value for each class, so that the probabilities all add up to 1.

$$\text{softmax} \longrightarrow f(x) = S(x)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{3.10}$$

#### Dropout

Dropout is a technique where we, during training, randomly drop a number of neurons from our network. This regularisation method has shown to be very successful at reducing overfitting; a common problem in machine learning tasks. We can tweak the effect of this method by altering the dropout value ([0-1]), which signifies what percentage of neurons we want to drop in a layer. More information about Dropout can be found in the original paper by Srivastava et al. [24].

#### Epochs

The number of epochs signifies how many times we pass our training data through our network for training.

#### Batch Size

When we are updating our model's weights after every data sample that we show it during training, it is called stochastic gradient descent. However, we don't have to update the model weights that often. We can also wait with updating the model weights until a certain amount of data samples (a batch) have been passed through our network. Then we calculate the loss *over the entire batch*, and adjust the weights accordingly. When the batch is as big as the entire training set, we call it batch gradient descent. If it is smaller, then we call it mini-batch gradient descent. Batch gradient descent is computationally more efficient, but is also slower, especially for big datasets. It is thus more common to use mini-batch gradient descent as a middle-ground between the other two.

### 3.2.4 Practical considerations

When training our NN, there are a few practical things we need to take into consideration. They relate to the training/validation/test-sets and how to search for the best hyperparameters.

#### Train/Validation/Test-split

When training a NN we split the dataset into three distinct sets; a training set, a validation set and a test set.

The training set is the data we feed to our model so that it can learn the underlying representation of the data. In other words, the training set is used to fine-tune the *weights* of our network. This is usually the biggest set.

The validation set is used to prevent our model from overfitting. After each training run, we *validate* our model against the validation set to verify that our model's performance is going in the right direction. If our model starts doing worse on the validation set than before, we have most likely *overfit* on our training data. This means that our model has learned more of the peculiarities of the training data, rather than the true underlying representation that we want to model.

The test set is the data we evaluate our model on, and it consists of data that the model has not been trained on, just like the validation set. For each sample in the test set, we let our model make a prediction, and then we evaluate how that went.

Rather than having one specific training and validation set (which might have some bias), it is also possible to utilise a process called $k$-fold cross-validation (see Figure 3.3). In cross-validation we start by creating a slightly bigger training set, which we then separate into $k$ (equally big) parts, so-called 'folds'. In each training run, one fold is withheld and used as a validation set, while the remaining folds are used for training the model. Afterwards, the performance is evaluated on the test set. This process is repeated $k$ times, upon which the average performance across the $k$ folds is calculated.

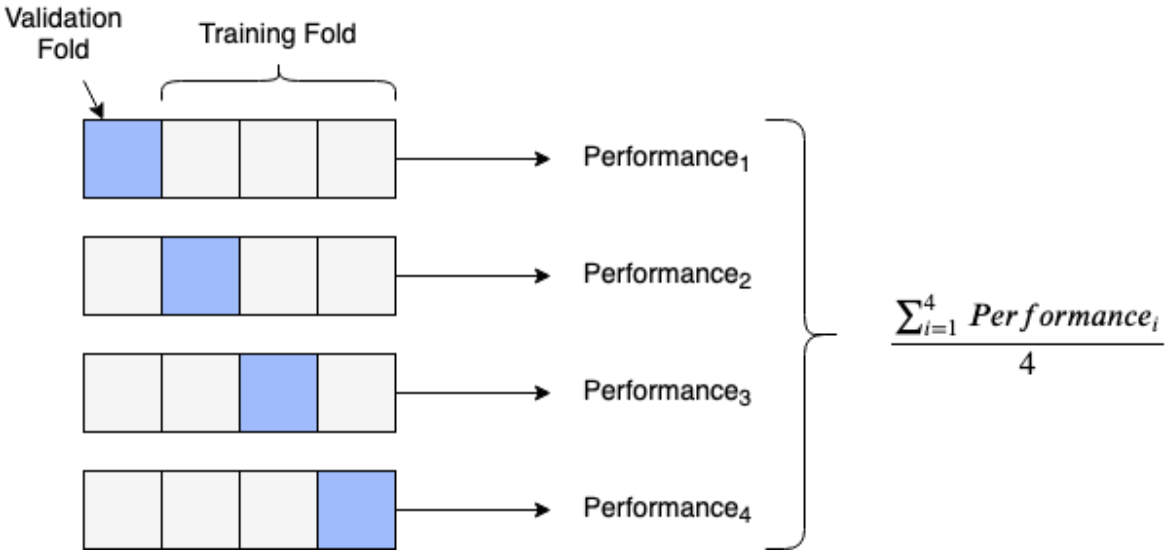



Figure 3.3: Visualisation of k-fold cross-validation. Adapted from a Figure by Ethen Liu [25].

**Searching for the best Hyperparameters**

Every Neural Network is different, and the underlying data that it is trying to represent is always unique for different problem settings. Hyperparameters that work well for one specific type of problem and/or dataset is not guaranteed to also perform well on others. There simply does not exist one unique set of hyperparameters that will always perform best. It is thus of importance to always tweak the hyperparameters of the NN for the specific problem one is trying to solve. It is possible to somewhat use ones intuition to estimate the range of certain hyperparameters, but not even experienced NN users are able to determine exactly the right settings beforehand.

Tweaking the hyperparameters of a neural network thus means that we train and evaluate a model using many different combinations of hyperparameter values, to find which combination works best. There are different methods of how to search for the best hyperparameters, but two common ones are grid search and random search.

In grid search we specify for each hyperparameter which exact values we want to test. We then train a model with every possible combination of hyperparameters. The number of simulations we have to do with grid search thus increases exponentially with the number of hyperparameters we want to test.

In random search, we can also specify for each hyperparameter which exact values we want to try with it. However, we can also specify a *range* of values. When running a simulation, our model will then randomly sample a hyperparameter value from the given range, using a specified distribution, e.g. uniform distribution if every value in the range should be equally likely. This allows us to test a wider variety of unique hyperparameter values, in fewer simulations. In cases where we want to test a wider range of hyperparameter combinations, it might thus be better to use random search when searching for the best set of hyperparameters.

# 4 Method

This chapter begins by detailing the datasets that were used in this thesis and how they were processed. Afterwards, an analysis of the data is presented, and the chapter concludes by detailing how the model was trained, tweaked and evaluated.

## 4.1 Data

The data used in this thesis were taken from three sources; Volvo Cars Corporation (VCC), the Swedish Meteorological and Hydrologist Institute (SMHI) and the Swedish Transport Administration (Trafikverket). The core data from VCC contained vehicle sensor data, while the data from SMHI and Trafikverket was treated as auxiliary road weather information.

### 4.1.1 Data from VCC

The data by VCC used in this thesis was collected during an expedition in northern Sweden, between the 11th and 14th of March, 2019. Data was collected during four days, but due to some complications, the data from the first day could not be used for this project. Data was collected both from private and public roads. Data was collected for roughly 8 hours each day, except for the 4th (and last) day where data was collected during less than 3 hours. Thus there are far fewer data samples from this day, which can also be seen in Table 4.1. The data from VCC consists of 90 features, which can all be seen in Appendix A.

Table 4.1: The VCC dataset used in this thesis. Each sample consists of measurements from 90 different features, such as for example Steering wheel angle, wheel speed, lateral and longitudinal forces on the wheels etc. Full list in Appendix A.

| **Day** | **Data Samples** |
|---|---|
| 2 | 2.1M |
| 3 | 1.8M |
| 4 | 0.6M |
| *Total* | 4.5M |

The main feature of interest in this thesis is naturally the measured road friction. The dataset provided by VCC contains two measured friction values, 'Friction_1' and 'Friction_2', measured on the left and right side of the vehicle. Either one can be used as the target feature, or as a combination of some sorts (e.g. min/max/average), but in this thesis, we have opted to focus on 'Friction_1'. Henceforth, when we are referring to the friction value, we are thus referring to 'Friction_1'.

### 4.1.2 Weather Data from SMHI

SMHI stands for ***S**veriges **m**eteorologiska och **h**ydrologiska **i**nstitut*, and it is a public Swedish agency collecting weather-related data and doing weather forecasts, among other things. SMHI has weather stations all around the country, collecting data such as Air Temperature, Air Humidity, Air Pressure, Precipitation, Wind and Sun Hours (see Appendix B). Through their API, it was possible to fetch the historical weather observations that were collected during the time of the expedition from nearby weather stations (see Figure 4.2).

The data from SMHI was supposedly collected once per hour for the most relevant features. However, there were some anomalies and several gaps in the data of many stations. In those cases, the missing data points were filled using the latest known values from the respective stations. This also aligns with what we would have to do if our system was already live and working, and a weather station stopped sending new measurement data.

### 4.1.3 Road Weather Data from Trafikverket

Trafikverket is the Swedish Transport Administration. Trafikverket's responsibilities lie, among other things, in "*building, operating and maintaining public roads and railways*", and all procedures related to driving licenses (in Sweden). As a part of that duty, they collect road weather data from their own weather stations all around the country. They enable access to this data through a service called VVIS (***V**ägväder**i**nformations**s**ystem*).

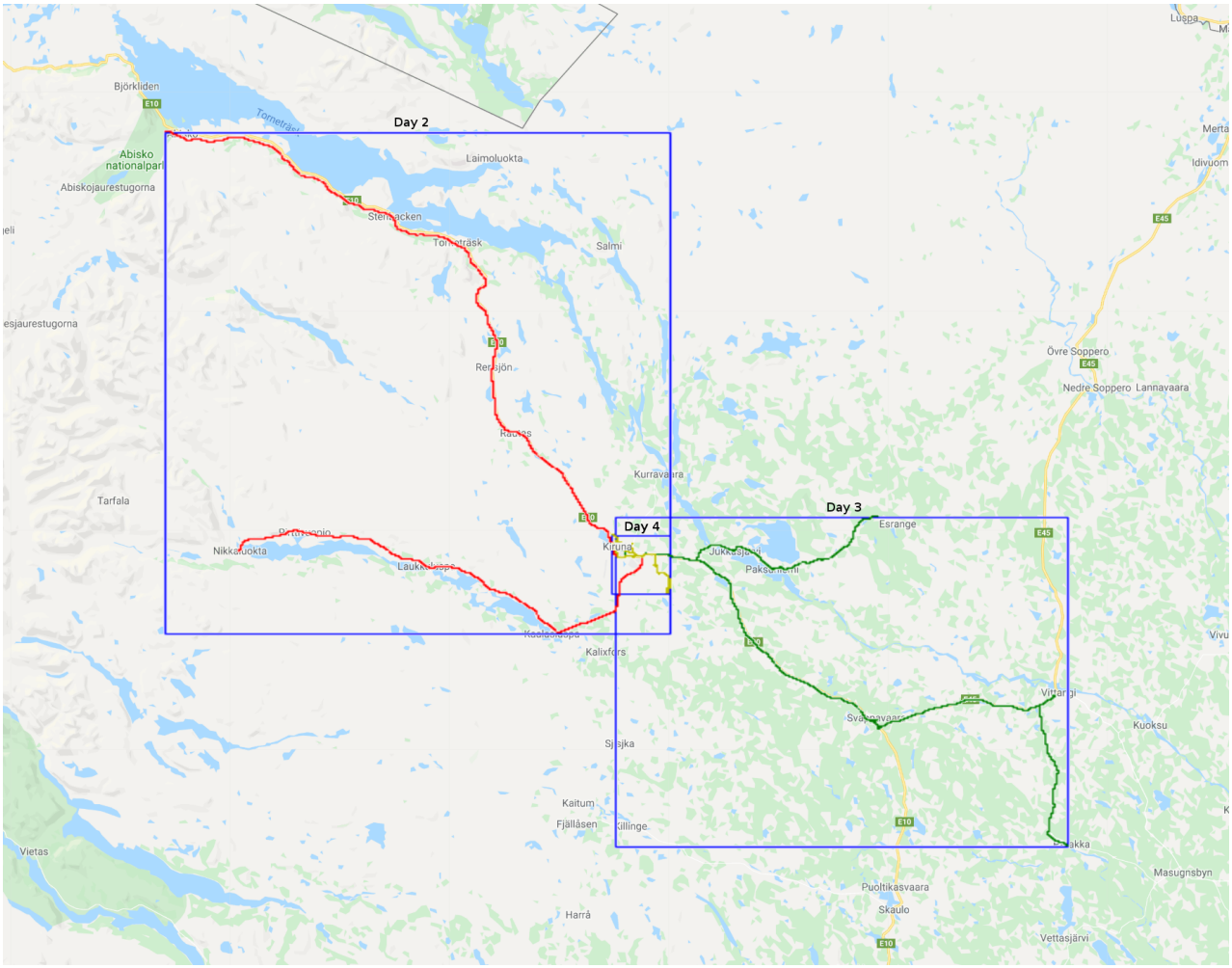



Figure 4.1: Map of where the data was collected. Red = Day 2, Green = Day 3, Yellow = Day 4.

Some of the data that is collected by Trafikverket is similar to the one provided by SMHI, but it also provides access to some more exclusive features, such as the road surface temperature (measured 2 mm above the road surface). The data from VVIS was acquired by an email request to Trafikverket. The stations that are closest to where the expedition took place can be seen in Figure 4.3. The measurements from Trafikverket's weather stations are logged once every 30 minutes, and the full set of features can be seen in Appendix C.

## 4.2 Data Analysis and Processing

In this section, we show how the data we had available looked like, and what steps we took to process them.

### 4.2.1 Data Distribution Analysis

In Figure 4.4 we can see how the friction value changes over small periods of time, and in Figure 4.5 we can see the friction distribution of the data at large. We can easily notice several things. Although the range of the friction value would normally go from 0 to 1, we can observe that the VCC data does not contain many measurements of road friction values higher than 0.5. This is because the measurement method that was used to collect the data was limited to friction values of at most 0.5, so a value of 0.5 could, in theory, represent any value in the range [0.5 - 1]. There are some outliers where higher friction values were recorded, but these were corrected to 0.5 for the sake of consistency during the latter parts of the project.

We can observe that there exist times when the friction is stable at 0.5, but that the measured friction values oscillate a lot. From Figure 4.5, we can also see that there is a huge spike at around the 0.5-mark,


Figure 4.2: Map of SMHI weather stations in the vicinity of where the expedition took place.


Figure 4.3: Map of Trafikverket's road sensor stations in the vicinity of where the expedition took place.

signifying that these higher values are a lot more common in our dataset than low friction values. This becomes a lot more obvious once we look at Figure 4.6, which showcases the same friction distribution, but with the friction binned into 6 classes; [0.0 - 0.1), [0.1 - 0.2) .. [0.4 - 0.5) and [0.5+]. Overall, slightly more than 67% of all samples show a friction value of 0.5. It is worth noting though that this differs between the days. This

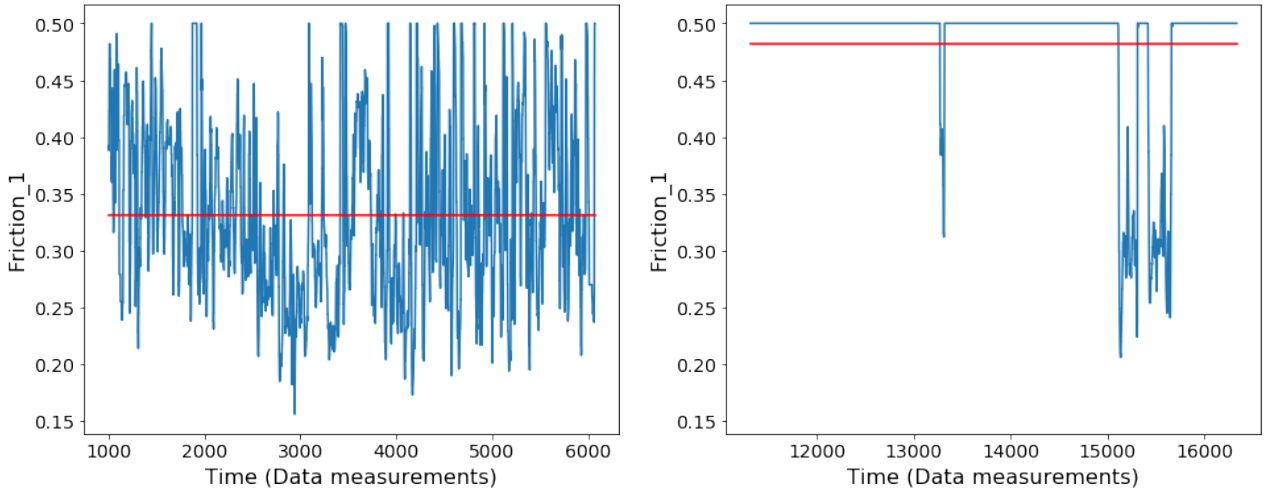



Figure 4.4: Friction distribution of the data over small periods of time. The red line showcases the average friction over the time periods.

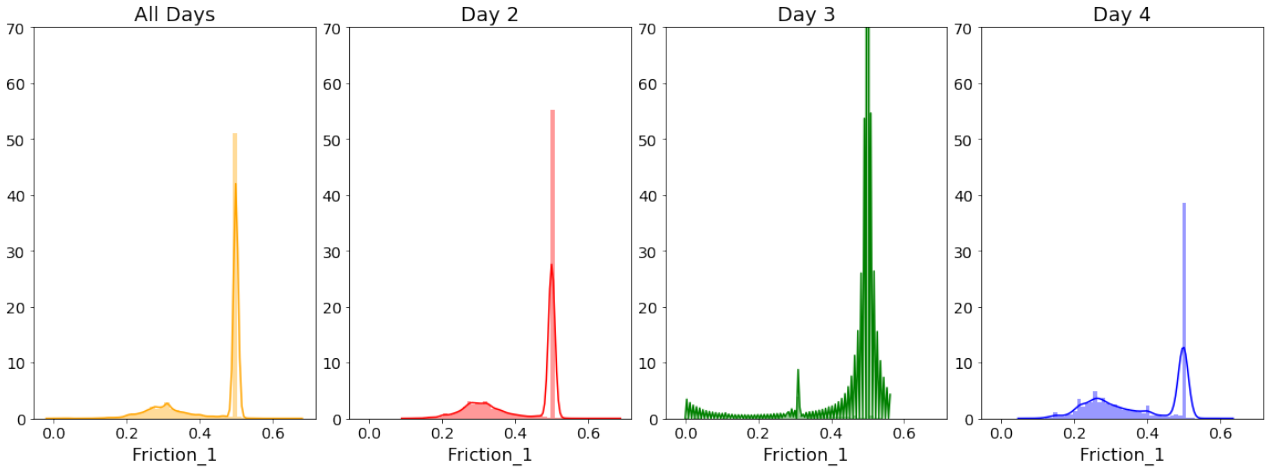



Figure 4.5: Friction distribution of the data.

fraction is over 80% for day 3, and only 41% on the last day, which, as we previously mentioned, also has the least amount of data samples. We will thus assume that the road conditions on the last day were significantly different from the other two days. This is also further supported by the weather data (see Appendix G), from which we can see that day 2 and 3 did not have snow, but that day 4 did.

In Appendix F and G we can find box plots with the distributions of all the VCC and weather features. We can observe that there are many features that only take on a single value. Similarly, there are also several binary features that almost exhibit the same characteristics, as one of the values appears in more than 99% of the cases. These are features that are very unlikely to help our model improve its performance significantly.

### Feature Correlation Analysis

A simple Pearson correlation analysis of the VCC features can be seen in Figure H.1 in Appendix H. We can see several clusters in the figure with intense colours, signifying groups of features with a stronger correlation. In some cases, the correlations are fairly logical, for example when certain features are being measured at each one of the vehicle's wheels.

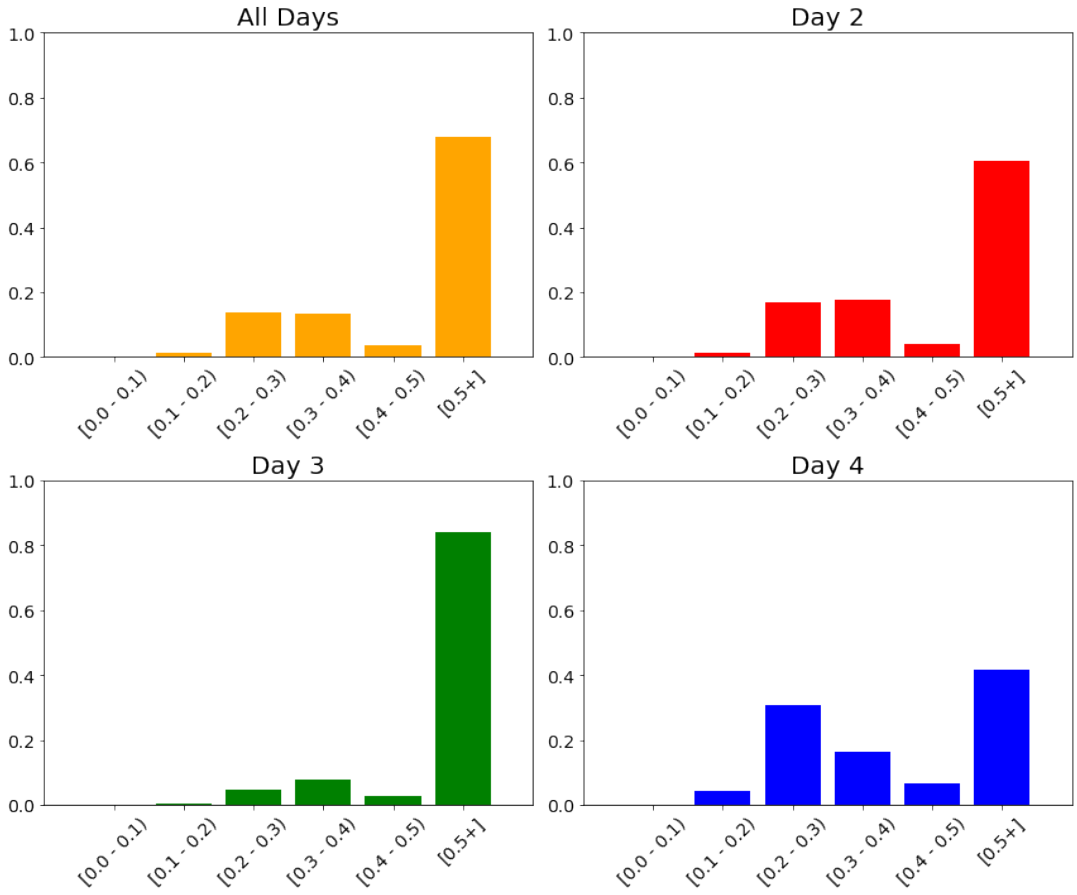



Figure 4.6: Friction distribution of the data, modelled as a classification problem and divided into 6 classes; [0.0 - 0.1), [0.1 - 0.2) .. [0.4 - 0.5) and [0.5+].

### 4.2.2 Feature Selection

A previous thesis by Chen [18] used a similar dataset from VCC as in this thesis. They selected a set of features based on the features' correlation and based on expert knowledge from the field of vehicle dynamics. After performing a correlation analysis on the dataset used in this thesis (see Section 4.2.1), as well as on the features used by Chen (see Figure H.2 in Appendix H), we decided to use the same in-vehicle features as proposed by Chen. The reasoning was two-fold: (1) it would be easier to compare the models in the future and (2) the selected features had comparably low correlations between each other.

The selection of weather-related features was made in a similar manner (based on a correlation analysis), and the outcomes of this are shown in Section 5.1. We settled on this specific way of selecting the features due to its simplicity, but there certainly exist more sophisticated methods. One such example would be to perform a Fast Correlation Based Feature Selection (FCBF), like Chen [18]. Another alternative is to perform a principal component analysis (PCA) to project the features into a lower-dimensional system. The drawback of the latter, however, is that the resulting projection is hard to interpret for a human, so we ultimately decided against doing so.

## 4.3 Experimental Setup

The core idea to assess whether road weather information has any added value for our RFE model, is to train three identical NN models that each make use of a different set of features: (1) one that is only using VCC features, (2) one that is only using road weather features, and (3) one that uses both VCC and road weather features combined. We also try this with many different hyperparameter settings, making sure that each NN model has a chance to find a suitable setting of hyperparameters that makes comparison fair. In order to do

this, we also first select a subset of weather-related features, as explained in Section 4.2.2.

Originally the plan was to test different model architectures on this task, such as a Random Forest, Neural Network and Long Short-Term Memory (LSTM) Network. However, our initial tests on the matter indicated that the proposed Neural Network slightly outperformed the other two, so we chose to focus on using a Neural Network in this project. The Neural Network architecture is furthermore a choice that several other research papers in the domain also decided upon (eg [14][15][16][26][17]). Although we made this decision, further investigating the usage of an LSTM network could be promising. On the other hand, the Random Forest performed quite bad in our initial test, to the extent that we concluded it unnecessary to further investigate the usage of it in this thesis.

### 4.3.1 Model Selection and Hyperparameter Optimisation

We used a simple, fully-connected NN with 3-4 hidden layers and an optional Dropout layer. An example of this architecture can be seen in Figure 4.7. The input layer represents the input features, and the number of neurons in this layer thus varies depending on the number of input features that were used. Likewise, the hidden layers all consist of $n$ neurons, depending on the current hyperparameter settings. Before being sent to the input layer, the input features were also scaled with Z-score standardisation (see Equation 3.2 on page 12). Lastly, the output layer consists of 6 neurons with a softmax activation function, one for every class of the binned target feature. Dropout may be applied, depending on the current hyperparameter setting, between the 2nd and 3rd hidden layer.

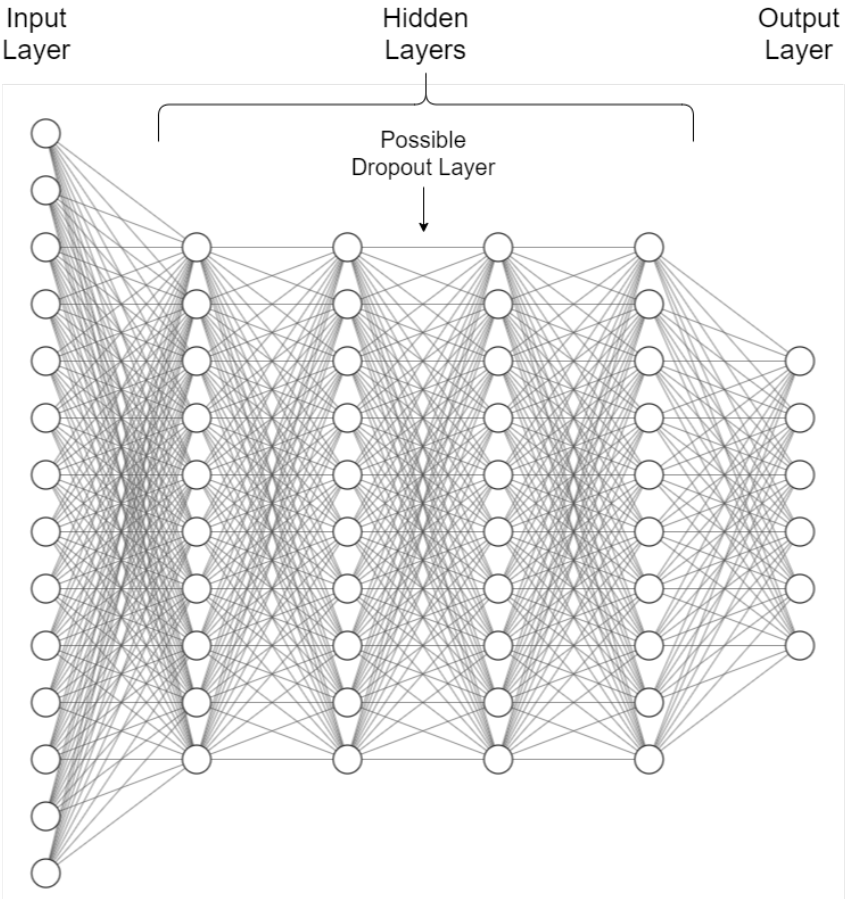



Figure 4.7: Model architecture. The input layer consists of as many neurons as there are input features, while the hidden layers have a different number of neurons, depending on the hyperparameter settings. The final output layer uses a softmax activation function.

The range of hyperparameters that were tested can be seen in Table 4.3.1. We followed a random search pattern, so not all possible hyperparameter combinations were tested. This would simply be too time-consuming.

Table 4.2: The range of tested hyperparameters, using a random search pattern.

| **Hyperparameter** | **Values** | **Sampling Distribution** |
|---|---|---|
| Neurons | [32 - 512] | Uniform (Integer) |
| Number of Hidden Layers | [3, 4] | - |
| Number of Layers with Dropout | [0, 1] | - |
| Dropout value | [0.0 - 0.5] | Uniform |
| Activation Function | ['sigmoid', 'relu', 'leaky relu'] | - |
| Leaky ReLu alpha | [0.0 - 0.5] | Uniform |
| Learning Rate | [0.0008 - 0.01] | Uniform |
| Epochs | [6, 8, 10, 12] | - |
| Batch Size | [1024, 2048, 4096] | - |

### 4.3.2 Train/Validation/Test-split

The data from day 2 and day 3 was used for the training set, while the data from day 4 was used for testing. A 4-fold cross-validation split was used to withhold a portion of the training set when training the model.

### 4.3.3 Evaluation

The classification models were mainly evaluated and optimized using the accuracy over all classes (see Equation 3.3 on Page 12) However, precision, recall and f1-scores were also logged, and were all used to better understand the real-world performance of the models. In particular, we paid attention to the classification report our model generated to see which classes it was able to predict well, and which ones it did not.

# 5 Results & Discussion

This chapter consists of two parts. First, we explain how we selected the most relevant weather features. We then present the results related to the question of what added value the road weather information brings, and how well our model performs.

## 5.1 Relevant road weather information features

We started by training both an RF and NN model on the dataset, utilising the previously selected features of the VCC dataset, as well as all 15 of the weather-related features. We trained these two models with a basic set of hyperparameters, but even after tweaking these hyperparameters for a bit, we were only able to beat the baseline performance by less than a single percentage point. Figuring that there was something wrong with some of the weather features, we proceeded to take a closer look and to make an attempt at reducing the dimensionality, i.e. by only keeping the relevant weather features.

Figure 5.1 shows the correlation between all weather features and the road friction in descending order, with the features that have the highest correlation with the road friction first. The feature with the highest linear correlation was "Vindmax m/s", with a Pearson coefficient of 0.26, and the feature with the least linear correlation was "TYta °C" (Ground Temperature, measured 2mm above the surface) with a Pearson coefficient of -0.00059. This contradicts one of our initial intuitions that the ground temperature would have a high correlation with the road friction.

Following this, we manually selected a set of the weather features, while trying to avoid redundant ones, i.e. features that are highly correlated with each other. Doing this we selected "Vindmax m/s", "TYta °C", "TLuft °C", "Snö mm" and "Global Irradians (svenska stationer)" as our initial weather features and tried to repeat the experiment from before. Doing so, we were now not even able to beat the baseline performance of 41.8% accuracy. Despite further switching one or two weather features, the performance barely improved, only sometimes beating the baseline, and only by a fraction of a percentage point (<1%).

We then proceeded with a slightly more practical and hands-on approach of testing a much wider variety of weather features - by simply testing many different weather feature combinations. Doing a full, exhaustive search with all possible weather feature combinations is not feasible, but the underlying idea is that even just trying a subset of combinations could give us better clues as to which weather features are of importance, and which ones are not. We thus repeated the experiments from before, this time using only the NN, and selecting a different combination of weather features each time, as well as a different number of them. In total, we performed 716 simulations with these settings.

By analysing the weather features that were part of the models that performed the best with this approach, we were able to select a more promising set of features, consisting of "Lufttemperatur", "Byvind", "TYta - Daggp °C", "Lufu %", "Solskenstid", "Daggpunktstemperatur" and "Relativ Luftfuktighet". What's noteworthy with this set of features is that many of them had a high correlation between each other, as can be seen in Figure 5.2. Although this set of features showed promising results for the dataset used in this thesis, it is uncertain how the same features would fare when using a completely different dataset. It is not that unlikely to imagine that a slightly different set of features might perform better when used with a different dataset. An overview of each weather features' impact on the model's performance can be seen in Figure D.1 in Appendix D.

## 5.2 Added value of road weather information

Given the settings proposed in Chapter 4, a neural network with a fixed architecture was trained on the given data with the selected features. The model was tested using many different combinations of hyperparameters, and every combination of hyperparameters was tested for each of the three feature sets; once using only the features from VCC, once using only the features from the weather, and once using all the features (VCC and weather combined). The results of those tests can be seen in Figure 5.3.

As can be seen in Table 5.1, the best performing model used the combined features from VCC and the weather. Its accuracy of 49.03% was 3.76 percentage points better than the best performing model when just using the VCC features. This represents an 8.3% improvement in terms of correctly classified samples. When taking all the 108 hyperparameter combinations into account, the average performance was significantly lower (4.93 percentage points) than our best performing set of hyperparameters. This shows that the right

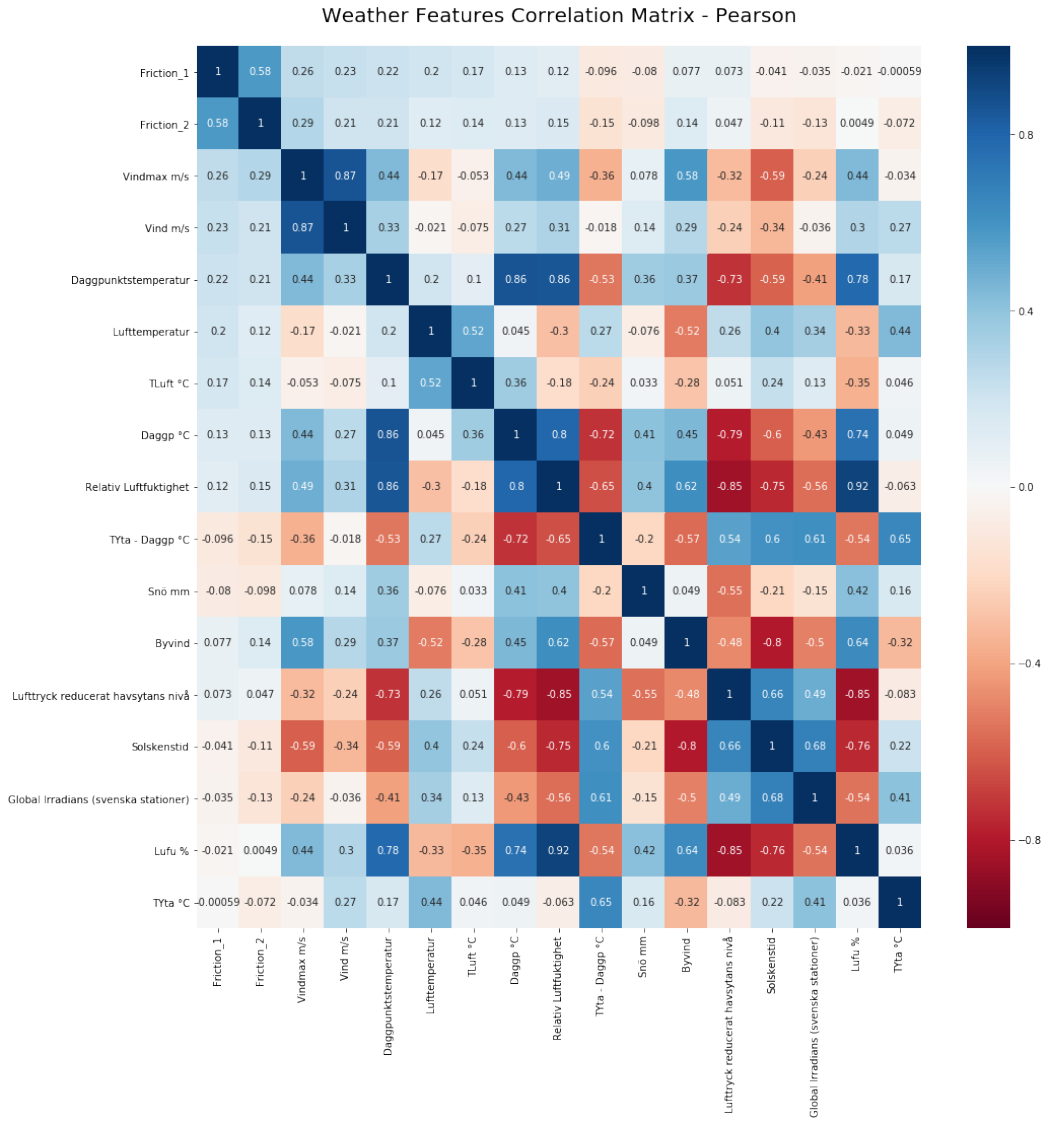



Figure 5.1: Correlation Matrix of the road weather features and the road friction variables, using the Pearson correlation coefficient. A value of 1 (blue) means a completely positive linear relationship between two features, while -1 (red) indicates a completely *negative* linear relationship. A value of 0 (white) signifies that no linear relationship exists at all.

hyperparameters also have a significant impact on the final result of the trained model. The hyperparameters of the best performing model can be found in Table 5.2.

Table 5.1: The model performance, using 108 different combinations of hyperparameters. The baseline is the performance we would get if we would always predict the majority-class on the specific dataset.

| | **All features** | **Only VCC features** | **Only weather features** | *Baseline* |
|---|---|---|---|---|
| *Best accuracy* | 49.03% | 45.27% | 41.82% | 41.82% |
| *Average accuracy* | 44.10% | 42.82% | 41.77% | - |
| *Average accuracy st dev* | 1.65% | 0.87% | 0.60% | - |
| *Validation set accuracy of best-performing model* | 81.6% | 76.4% | 73.7% | 71.5% |

As can be seen in Table 5.3, the best performing model was fairly good at estimating the road friction

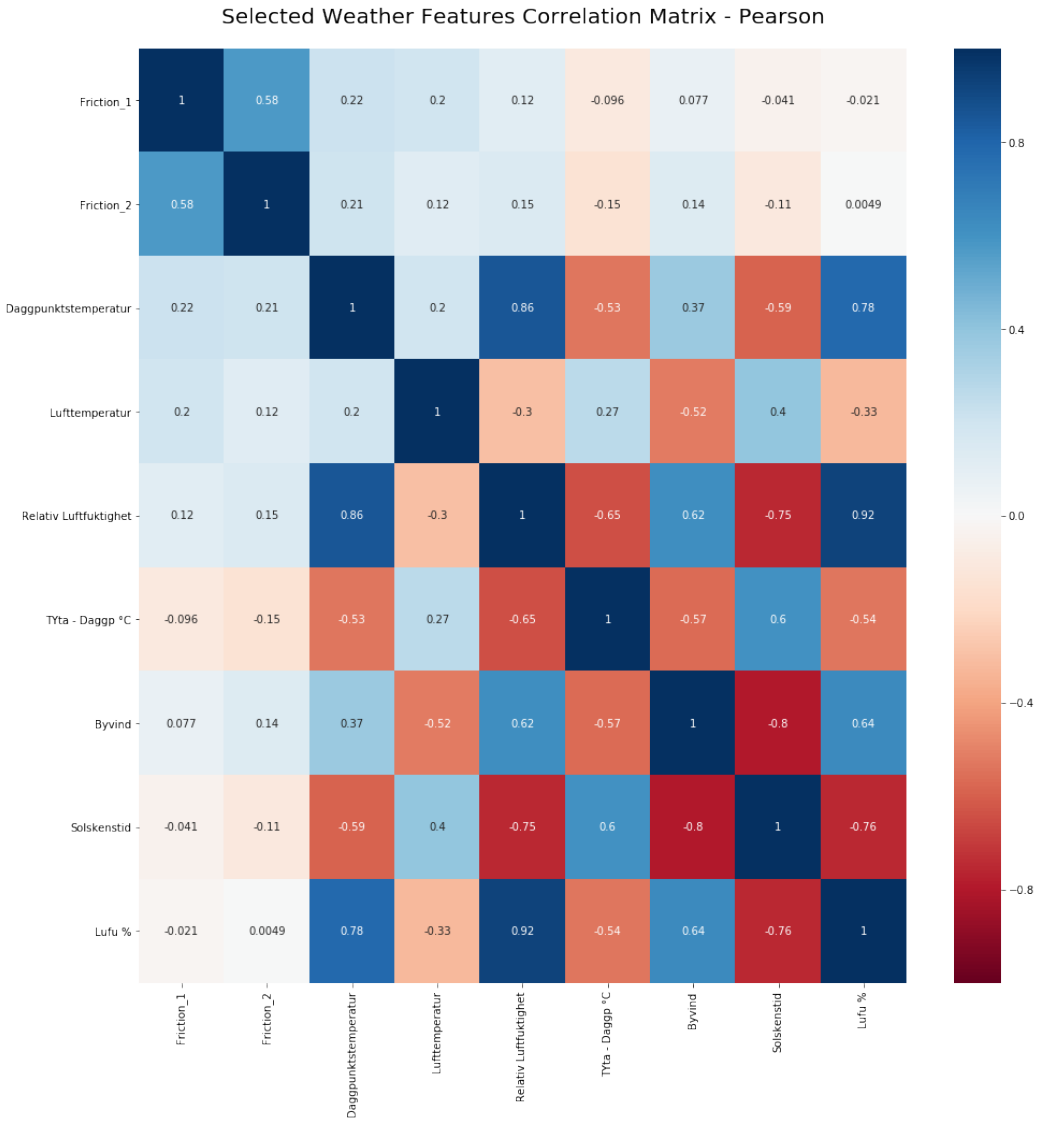



Figure 5.2: Correlation Matrix of the selected road weather features and the road friction variables, using the Pearson correlation coefficient. A value of 1 (blue) means a completely positive linear relationship between two features, while -1 (red) indicates a completely *negative* linear relationship. A value of 0 (white) signifies that no linear relationship exists at all.

when the road friction was 0.500 and between 0.200 - 0.299. This can also be seen from the model's confusion matrix, which can be found in Appendix E. However, it's important to note that the model often predicted a friction value of 0.500, which can be understood based on the high recall for that class. This can also easily be explained by the fact that friction values of 0.500 made up over 67% of the training data. Meanwhile, the performance for lower friction bins was a lot worse and it is clear that the model's estimation capabilities in the lower friction ranges still could improve a lot.

### 5.2.1 Limitations

It was already pointed out in the previous section that our best performing model predicted the friction class of 0.500 in many cases when it should not, and that it did improve the performance. However, this gives rise to the question of where the model's improved performance due to the added weather data is actually coming from. We can get a glimpse of this by looking at the classification report of the best performing model that was using only the VCC features in Table 5.4. While the precision values are roughly on similar levels, it's evident that the recall values are lower across the board, apart from the 0.500 class that went from 0.958 to

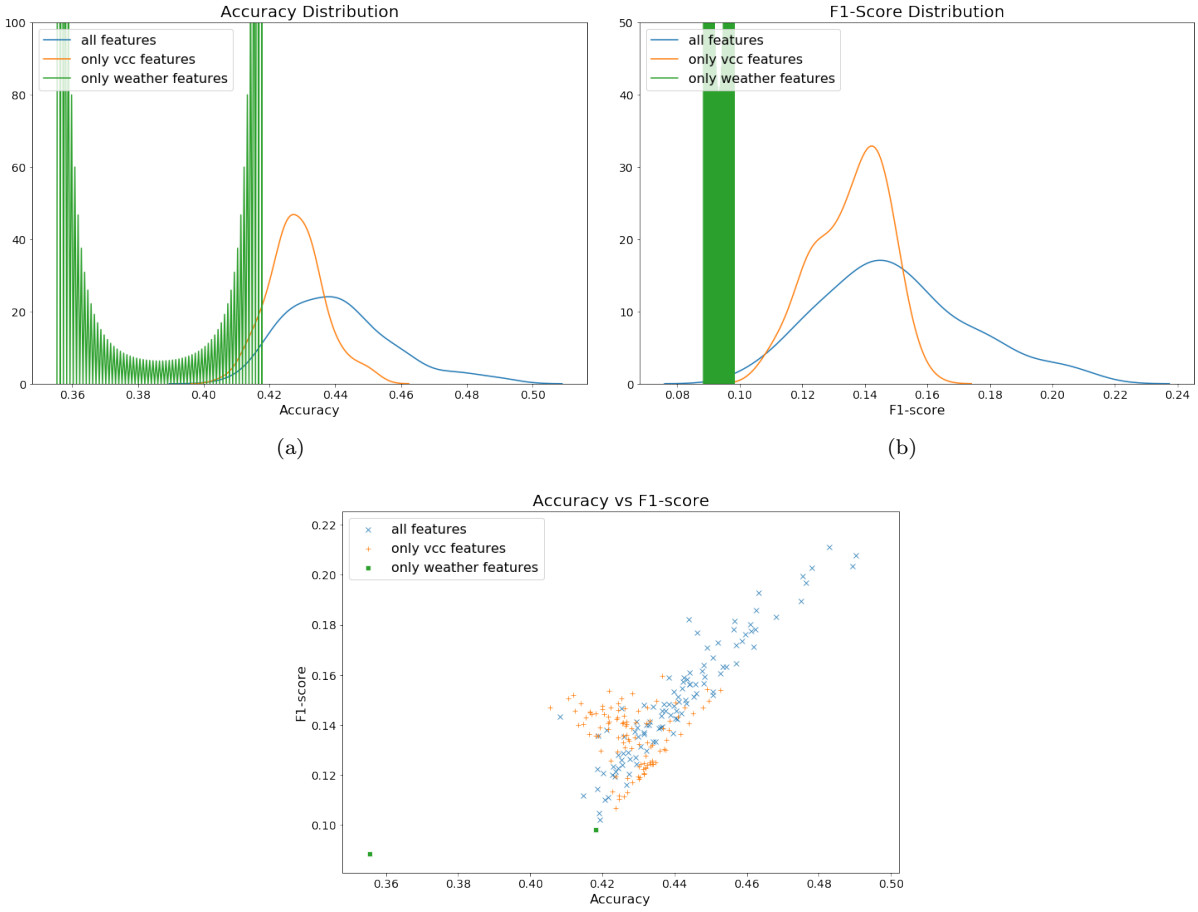



Figure 5.3: Results from the the initial tests, showcasing: (a) the accuracy distribution; (b) the F1-score distribution; and, (c) how accuracy compares to F1-score;

0.99. This means that we correctly predict 99% of all the friction values that were supposed to be in the 0.500 class. Thus, it looks as if we can draw the conclusion that the added weather features help our model to better generalise by not predicting a friction class of 0.500 as often.

However, to further investigate where the added value really comes in, we chose to also test our model's predictive capabilities when dealing with binary target values, as well as target values when not taking the friction class of 0.500 into consideration.

### Binary classification

The results of the same test being done as a binary classification task can be seen in Figure 5.4. As can be seen in Table 5.5, the best performing model here also used the combined features from VCC and the weather data. Its accuracy of 75.97% was 21.5 percentage points better than the best performing model when just using the VCC features. This represents a 39.5% improvement in terms of correctly classified samples. However, it's worth noting here that the majority class of the *test*-set when modelling as a binary classification problem is not 0.500 anymore (but still is in the training set). By always predicting that the friction is not 0.500 one should thus be able to achieve a 58.8% accuracy. That the model using only VCC features is not able to beat this performance, while using all features can, further proves that the added weather features are indeed useful to better predict the friction in the lower ranges.

As can be seen in Table 5.6, the best performing model was fairly good at estimating the road friction when the road friction was divided into only two classes. Comparing this to the classification report of the best performing model using only the VCC features in Table 5.7, we can based on the recall values again see that our model without the added weather tries to predict a friction value of 0.500 way more often than it

Table 5.2: The hyperparameters of the best performing model.

| **Hyperparameter** | **Values** |
|---|---|
| Neurons | 128 |
| Number of Hidden Layers | 4 |
| Number of Layers with Dropout | 0 |
| Dropout value | Not applicable. |
| Activation Function | Leaky ReLU |
| Leaky ReLu alpha | 0.4 |
| Learning Rate | 0.002 |
| Epochs | 10 |
| Batch Size | 4096 |

Table 5.3: Classification report of the best performing model. An alternative representation of this data can be found in the form of a confusion matrix in Appendix E.

| | **precision** | **recall** | **F1-score** | **support** |
|---|---|---|---|---|
| *[0.0 - 0.1)* | 0.000 | 0.000 | 0.000 | 48 |
| *[0.1 - 0.2)* | 0.360 | 0.105 | 0.145 | 8,056 |
| *[0.2 - 0.3)* | 0.538 | 0.190 | 0.258 | 55,597 |
| *[0.3 - 0.4)* | 0.390 | 0.167 | 0.165 | 29,919 |
| *[0.4 - 0.5)* | 0.048 | 0.003 | 0.005 | 11,718 |
| *[0.5+]* | 0.525 | 0.958 | 0.675 | 75,729 |
| **accuracy** | | | 0.49 | 181,067 |
| **macro avg** | 0.31 | 0.24 | 0.21 | 181,067 |
| **weighted avg** | 0.08 | 0.08 | 0.07 | 181,067 |

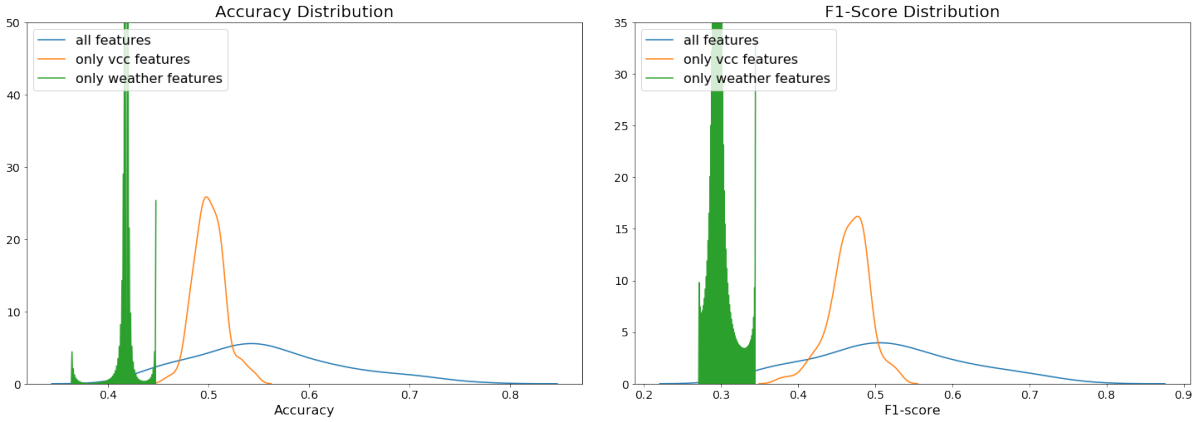
Table 5.4: Classification report of the best performing model, using only the VCC features.

| | **precision** | **recall** | **F1-score** | **support** |
|---|---|---|---|---|
| *[0.0 - 0.1)* | 0.00 | 0.00 | 0.00 | 48 |
| *[0.1 - 0.2)* | 0.29 | 0.08 | 0.12 | 8,056 |
| *[0.2 - 0.3)* | 0.53 | 0.10 | 0.16 | 55,597 |
| *[0.3 - 0.4)* | 0.39 | 0.02 | 0.04 | 29,919 |
| *[0.4 - 0.5)* | 0.05 | 0.01 | 0.02 | 11,718 |
| *[0.5+]* | 0.45 | 0.99 | 0.63 | 75,729 |
| **accuracy** | | | 0.45 | 181,067 |
| **macro avg** | 0.28 | 0.20 | 0.16 | 181,067 |
| **weighted avg** | 0.07 | 0.08 | 0.05 | 181,067 |

Table 5.5: The model performance for binary classification, using 99 different combinations of hyperparameters. The baseline is the performance we would get if we would always predict the majority-class on the specific dataset.

| | **All features** | **Only VCC features** | **Only weather features** | *Baseline* |
|---|---|---|---|---|
| *Best accuracy* | 75.97% | 54.47% | 44.77% | 58.2% |
| *Average accuracy* | 55.77% | 50.09% | 41.80% | - |
| *Average accuracy st dev* | 7.26% | 1.51% | 0.62% | - |
| *Validation set accuracy of best-performing model* | 88.8% | 83.4% | 83.9% | 71.5% |

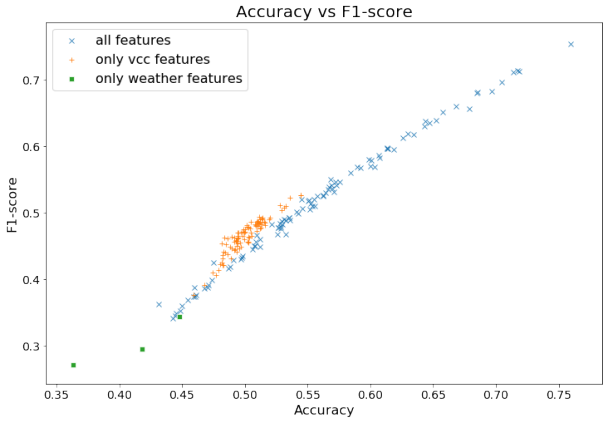
should. This again proves that the added weather features provide us with a significant added value in terms of differentiating low from high friction values.




(a)




(b)




(c)

Figure 5.4: Results from the test on binary classification, showcasing: (a) the accuracy distribution; (b) the F1-score distribution; and, (c) how accuracy compares to F1-score;

Table 5.6: Classification report of the best performing binary classification model. An alternative representation of this data can be found in the form of a confusion matrix in Appendix E.

| | **precision** | **recall** | **F1-score** | **support** |
|---|---|---|---|---|
| *[0.0 - 0.5)* | 0.83 | 0.75 | 0.78 | 105,338 |
| *[0.5+]* | 0.70 | 0.78 | 0.73 | 75,729 |
| | | | | |
| **accuracy** | | | 0.76 | 181,067 |
| **macro avg** | 0.76 | 0.76 | 0.75 | 181,067 |
| **weighted avg** | 0.39 | 0.38 | 0.38 | 181,067 |

Table 5.7: Classification report of the best performing binary classification model, when only using VCC features.

| | **precision** | **recall** | **F1-score** | **support** |
|---|---|---|---|---|
| *[0.0 - 0.5)* | 0.95 | 0.26 | 0.37 | 105,338 |
| *[0.5+]* | 0.49 | 0.97 | 0.65 | 75,729 |
| | | | | |
| **accuracy** | | | 0.56 | 181,067 |
| **macro avg** | 0.72 | 0.61 | 0.51 | 181,067 |
| **weighted avg** | 0.38 | 0.28 | 0.24 | 181,067 |

**Using only low friction values**

Replicating the same experiments, but using only friction values below 0.500 (and above 0.100, as values below 0.100 almost do not appear at all) gives the results as shown in Figure 5.5. As can be seen in Table 5.8, the best performing model here actually only used the weather features. What's additionally noteworthy is that the hyperparameter combination that produced this result is the only one among 514 that barely managed to beat the baseline performance of always predicting the [0.2 - 0.3) class, which would result in a 52.8% overall accuracy. So while this was indeed the model with the highest performance in terms of accuracy, it can be argued that the model didn't actually learn much as it always predicted the same value (see Table 5.9). At the same time, the best model using VCC features had a significantly lower accuracy (46.74%), but a slightly higher F1-score (see Table 5.10), signifying that its performance is slightly better as an average across the board.

Table 5.8: The model performance when not using friction values of 0.500, using 514 different combinations of hyperparameters. The baseline is the performance we would get if we would always predict the majority-class on the specific dataset.

| | **All features** | **Only VCC features** | **Only weather features** | *Baseline* |
|---|---|---|---|---|
| *Best accuracy* | 52.73% | 46.74% | 52.89% | 52.8% |
| *Average accuracy* | 26.31% | 40.80% | 25.43% | - |
| *Average accuracy st dev* | 10.08% | 1.95% | 12.04% | - |
| *Validation set accuracy of best-performing model* | 63.8% | 57.9% | 56.1% | 46.1% |

Table 5.9: Classification report of the best performing model, when not using friction values of 0.500.

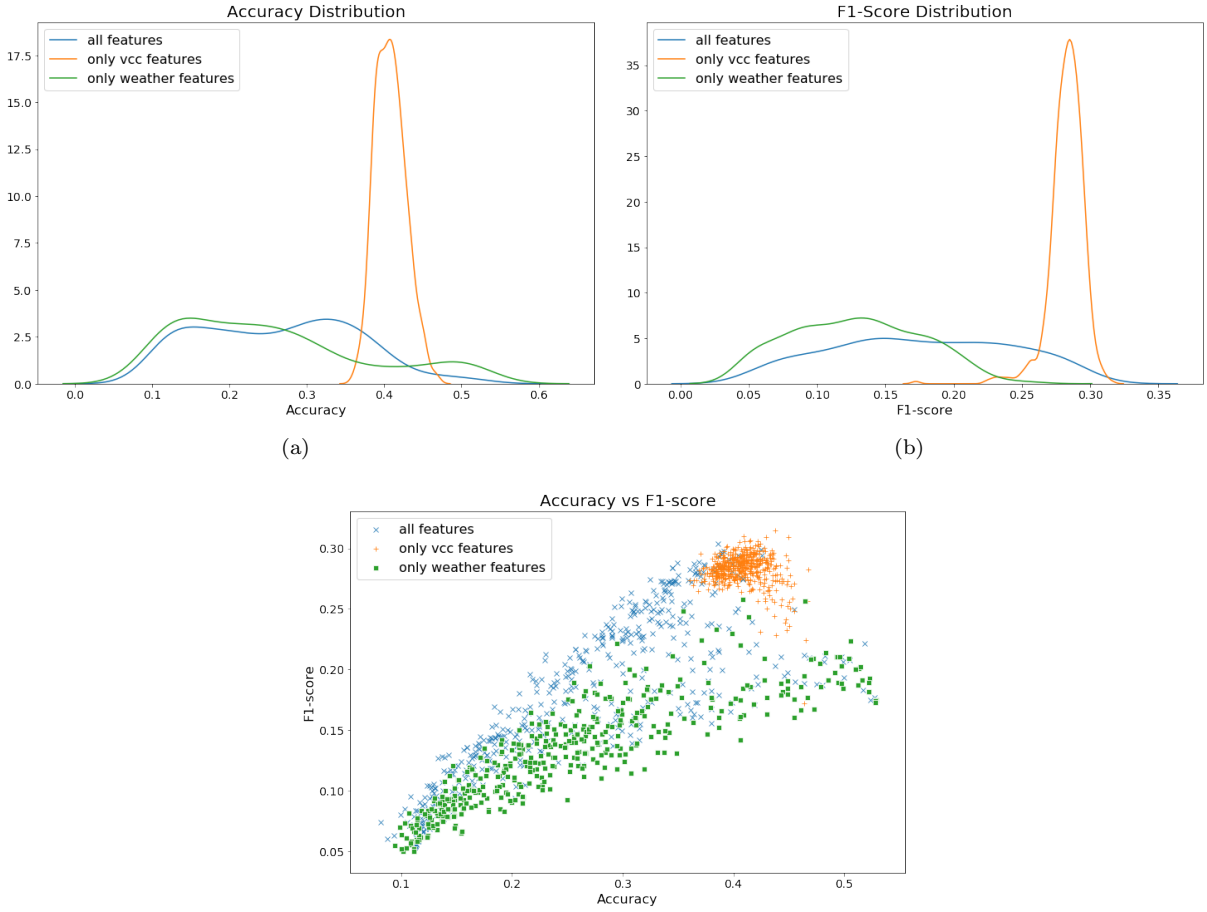
| | **precision** | **recall** | **F1-score** | **support** |
|---|---|---|---|---|
| *[0.1 - 0.2)* | 0.04 | 0.00 | 0.00 | 8,040 |
| *[0.2 - 0.3)* | 0.53 | 0.99 | 0.69 | 55,597 |
| *[0.3 - 0.4)* | 0.51 | 0.01 | 0.01 | 29,919 |
| *[0.4 - 0.5)* | 0.09 | 0.00 | 0.00 | 11,718 |
| **accuracy** | | | 0.53 | 105,274 |
| **macro avg** | 0.27 | 0.25 | 0.18 | 105,274 |
| **weighted avg** | 0.11 | 0.13 | 0.09 | 105,274 |

Table 5.10: Classification report of the best performing model, when not using friction values of 0.500 and using only the VCC features.

| | **precision** | **recall** | **F1-score** | **support** |
|---|---|---|---|---|
| *[0.1 - 0.2)* | 0.09 | 0.10 | 0.06 | 8,040 |
| *[0.2 - 0.3)* | 0.52 | 0.83 | 0.64 | 55,597 |
| *[0.3 - 0.4)* | 0.47 | 0.05 | 0.09 | 29,919 |
| *[0.4 - 0.5)* | 0.06 | 0.04 | 0.05 | 11,718 |
| **accuracy** | | | 0.47 | 105,274 |
| **macro avg** | 0.28 | 0.26 | 0.21 | 105,274 |
| **weighted avg** | 0.11 | 0.12 | 0.09 | 105,274 |

### 5.2.2 Discussion

When comparing the performance of our model against that of other research papers' (see Table 2.3 and 2.2 in Chapter 2, pages 7 and 8), it is apparent that our final performance is slightly unsatisfactory. Most other papers report performances of above 80%, whereas ours was only able to achieve 76% in the binary classification case (the <50% performance of our initial model is harder to compare with, since we chose to bin our target feature into six classes, rather than two or three like most other papers). This lack of performance can likely be




(a) (b)

(c)

Figure 5.5: Results from the test on classification, excluding friction values of 0.5, showcasing: (a) the accuracy distribution; (b) the F1-score distribution; and, (c) how accuracy compares to F1-score;

associated with our specific dataset, and how we split it up into a training and test set.

Looking back at Tables 5.1, 5.5 and 5.8, we can observe that our models' accuracies on the validating sets are all much higher than on our test sets. Although this is expected due to their different distributions (the test set had a baseline of 41.8%, compared to 71.5% on the validation set), it is clear that choosing a different split would have produced a model with a much higher performance on paper. Doing so would allow us to properly *compete* with the performances of others. However, this would not be a true sign of a better model, as it would be unlikely for our model to have better learnt to generalise this performance unto completely new data. Rather, it would have only learnt to better predict the data that was used in this thesis. As the ultimate goal of our RFE model is to be able to estimate the road friction on the roads *in real-life*, it would thus be misleading to change the training and test set splits in order to reach a better performance (on paper). It is obvious that the mismatch in distributions between the training and validation set, compared to the test set, is one of many potential sources for the lacking performance. It would perhaps have been possible to explore certain strategies to deal with this, such as over- or under-sampling certain classes, but this is not something that was properly addressed in this thesis. Hence, this could be something that is worth addressing in the future.

The volatility of the friction data is another potential source for the lacking performance. As can be seen in Figure 4.4 (page 21 in Chapter 4), the recorded friction values jump a lot between higher and lower friction values, appearing to be fairly random. Although the friction measurements are deemed to be very accurate, it would be interesting to smoothen the measured friction values, for example, by passing them through a Kalman filter.

# 6 Conclusion and Future Work

This chapter highlights the main conclusions, some ethical considerations and gives some insights into what could be interesting to investigate in the future.

## 6.1 Conclusion

It is shown that adding road weather information can indeed have a positive influence on the road friction estimation. When modelled as supervised classification problem with 6 classes ([0.0 - 0.1), [0.1 - 0.2) .. [0.5+]), training a model using the selected VCC features together with 7 of the weather features, compared to only training the model with the selected VCC features, resulted in an average improvement per model/set of hyperparameters of 1.28 percentage points (or 3.0% improvement in terms of correctly classified samples). The best performing model using the specific model configuration used in this thesis was even 3.76 percentage points better than the best performing model that only used VCC features (49.03% compared to 45.27% - an 8.3% improvement in terms of correctly classified samples).

When modelling the problem as a binary classification task ([0 - 0.5) and [0.5+]), we are able to achieve even bigger improvements when using the same setup. Here, the added weather features are able to improve our model's performance from 54.47% to 75.97% - a 21.5 percentage points increase (or 39.5% improvement in terms of correctly classified samples).

Taking into account our results from testing different binning configurations, we can thus show that the road weather information mainly helps to distinguish high from low friction, and vice-versa. However, it did not help the model very much to distinguish between different levels of low friction values.

We also conclude that the chosen training and test split used in this thesis results in a lower final performance compared to what other research papers achieved. While choosing a different split could help improve our performance on paper, it is unknown whether this would actually result in an improved *real-life* performance on the road.

## 6.2 Ethical considerations

There are no obvious ethical dilemmas with the simple act of trying to improve the road friction estimation performance of vehicles. In the ideal case, it should help the driver to be more careful in slippery road conditions and improve the vehicle's stability and overall performance under these circumstances. However, depending on how one implements it, there are certain caveats.

While one might want to believe that an ML-powered solution will perform better, which it often does, it also comes at a price. Usually, this is shown in cases where the model encounters something it has not seen before and thus hasn't had the opportunity to learn about. Unlike pure mathematical models, or other rule-based methods, an ML model's performance can be quite unpredictable under these circumstances. They also can't be "fixed" as easily, by manually adding a rule to the algorithm. The consequences that a bad estimation can have in such a situation depends mostly on how the model's estimations are used in the vehicle.

Let us assume the road friction estimations are used to give the driver a warning when they encounter slippery road conditions. If the estimations are often wrong, it will decrease the trust the driver has in the system, thus making it more likely for them to turn off the feature (if possible), or to simply disregard the warnings. Although one could argue that we're then back in the same situation as before we started giving out warnings, this interaction with the system could potentially have a slight influence on the driver's state of mind, as we tend to get irritated when something does not work as intended. It's also possible that the driver's trust in other safety functions of the vehicle decreases. As such, vehicle manufacturers should make sure that they are confident in their predictions, before passing on the information to the driver.

If the estimations are instead used in some of the security-critical AD components, it could arguably be even more important that we are confident in our road friction estimations. Otherwise, we might end up in situations where the RFE algorithm worsens the stability and performance of the vehicle. Of course, what type of effect this will have on the final outcome (e.g. if the vehicle will have a crash) is almost impossible to predict.

Finally, a potential security concern is what could happen if the system got hacked. A potential outcome in such a case could be that the system consistently gives (confident) road friction estimations that are wrong. The effects of that could be the same as mentioned above. However, it is our opinion that interfering with a

vehicle's RFE algorithm in this way is quite unlikely. If someone wanted to do harm and they were able to hack the system, there are other vehicle functions such as steering and braking that could have a far more devastating impact.

## 6.3 Future Work

The perhaps most promising thing that should be done for future tests is to gather a more varied dataset. Although the data from VCC contained many data samples, it was gathered during only three consecutive days so that the road and weather conditions were fairly consistent. To fully capitalise on the gains that the added road weather information could have, we would need to train a model with data that covers a wider range of existing road and weather conditions. This means that the data collection period would need to happen during a longer period of time, for example by collecting data during 100 consecutive days or more (but it should be sufficient to collect data from <1h of driving from each of these days).

When working with the data, it would also be worth investigating if smoothening the measured friction values could improve the performance. In the dataset that was used in this thesis, the measurements appeared to be fairly volatile. Applying a Kalman filter or similar could perhaps help in this aspect. Similarly, it could be worth investigating if over- or under-sampling could be used to deal with the friction class imbalance.

For research purposes, it would also be of interest to establish an open-source dataset that researchers can use to benchmark their algorithms and models. It is currently very hard, or near impossible, to properly compare the performances of different research papers, as they all use different datasets with different features. However, we are also aware that most of the existing datasets are proprietary and potentially contain information (such as features) that could be seen as confidential in some sort of way. We are thus not able to assess how willing the owners of these datasets are to openly share them. Ideally, one would also need to jointly agree on what error metric is the most important, although it would also be possible to skip this step by simply having all models report on a wide variety of error metrics. An idea could be to create something similar to ImageNet [27], but for road friction estimation (for more information about ImageNet, see the cited paper).

Architecture-wise, it would be very interesting to properly examine the performance of LSTM networks. This holds especially true when modelling the data as a *time-series*, where each data sample that the model is trained with also contains information from some of the previous measurements. Additionally, vehicle sensor data is prone to being noisy, and modelling the data as a time-series ought to help deal with some of that noise.

Slightly altering how the problem is framed, would also be something worth investigating. For example, rather than having the model estimate the friction, one could have it estimate the *change* in friction from one time-step to another. This is something that is often being done in stock price forecasting examples. Another example of re-framing the problem could be to have the model estimate the road friction *further ahead in time*, similar to what Panahandeh et al. did [17], or just a few hundred meters *further ahead on the road*. This could potentially allow the vehicle to warn the drivers earlier about potentially slippery road conditions. Finally, one could also model the task as a regression problem, i.e. estimating the exact road friction value rather than estimating that the road friction belongs to a specific class such as, e.g. [0.1-0.2).

# References


[1] S. Khaleghian, A. Emami, and S. Taheri. A technical survey on tire-road friction estimation. *Friction* **5**.2 (June 2017), 123–146. ISSN: 2223-7690, 2223-7704. DOI: 10.1007/s40544-017-0151-0. URL: http://link.springer.com/10.1007/s40544-017-0151-0 (visited on 07/01/2019).

[2] M. Acosta, S. Kanarachos, and M. Blundell. Virtual tyre force sensors: An overview of tyre model-based and tyre model-less state estimation techniques. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* **232**.14 (Dec. 2018), 1883–1930. ISSN: 0954-4070, 2041-2991. DOI: 10.1177/0954407017728198. URL: http://journals.sagepub.com/doi/10.1177/0954407017728198 (visited on 07/28/2019).

[3] M. Acosta, S. Kanarachos, and M. Blundell. Road Friction Virtual Sensing: A Review of Estimation Techniques with Emphasis on Low Excitation Approaches. *Applied Sciences* **7**.12 (Nov. 2017), 1230. ISSN: 2076-3417. DOI: 10.3390/app7121230. URL: http://www.mdpi.com/2076-3417/7/12/1230 (visited on 07/14/2019).

[4] R. Ghandour et al. "Tire/road friction coefficient estimation applied to road safety". *18th Mediterranean Conference on Control and Automation, MED'10*. Marrakech, Morocco: IEEE, June 2010, pp. 1485–1490. ISBN: 978-1-4244-8091-3. DOI: 10.1109/MED.2010.5547840. URL: http://ieeexplore.ieee.org/document/5547840/ (visited on 07/01/2019).

[5] S. Roychowdhury et al. "Machine Learning Models for Road Surface and Friction Estimation using Front-Camera Images". *2018 International Joint Conference on Neural Networks (IJCNN)*. Rio de Janeiro: IEEE, July 2018, pp. 1–8. ISBN: 978-1-5090-6014-6. DOI: 10.1109/IJCNN.2018.8489188. URL: https://ieeexplore.ieee.org/document/8489188/ (visited on 06/28/2019).

[6] Andersson, M et al. *Road friction estimation Part II*. Tech. rep. Nov. 2010, p. 32.

[7] S. Müller, M. Uchanski, and K. Hedrick. Estimation of the Maximum Tire-Road Friction Coefficient. *Journal of Dynamic Systems, Measurement, and Control* **125**.4 (2003), 607. ISSN: 00220434. DOI: 10.1115/1.1636773. URL: http://DynamicSystems.asmedigitalcollection.asme.org/article.aspx?articleid=1410132 (visited on 08/11/2019).

[8] F. Gustafsson. Slip-based tire-road friction estimation. *Automatica* **33**.6 (June 1997), 1087–1099. ISSN: 00051098. DOI: 10.1016/S0005-1098(97)00003-4. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109897000034 (visited on 07/14/2019).

[9] C. S. Ahn. "Robust Estimation of Road Friction Coefficient for Vehicle Active Safety Systems". PhD Thesis. University of Michigan, 2011.

[10] R. Rajamani et al. Algorithms for Real-Time Estimation of Individual Wheel Tire-Road Friction Coefficients. *IEEE/ASME Transactions on Mechatronics* **17**.6 (Dec. 2012), 1183–1195. ISSN: 1083-4435, 1941-014X. DOI: 10.1109/TMECH.2011.2159240. URL: http://ieeexplore.ieee.org/document/5959213/ (visited on 07/14/2019).

[11] C. Lex. "Maximum Tire-Road Friction Coefficient Estimation". Dissertation. Graz Technical University, 2015.

[12] Y.-Q. Zhao et al. Estimation of Road Friction Coefficient in Different Road Conditions Based on Vehicle Braking Dynamics. *Chinese Journal of Mechanical Engineering* **30**.4 (July 2017), 982–990. ISSN: 1000-9345, 2192-8258. DOI: 10.1007/s10033-017-0143-z. URL: http://link.springer.com/10.1007/s10033-017-0143-z (visited on 07/01/2019).

[13] Magnusson et al. "Real-time high-resolution road condition map for the EU". *9th International Munich Chassis Symposium 2018*. Munich: Springer Fachmedien Wiesbaden, 2019, pp. 851–875. ISBN: 978-3-658-22050-1. DOI: https://doi-org.proxy.lib.chalmers.se/10.1007/978-3-658-22050-1_56.

[14] W. R. Pasterkamp and H. B. Pacejka. Application of Neural Networks in the Estimation of TireIRoad Friction Usirig the Tire as Sensor (Feb. 1997), 9. DOI: 10.4271/971122.

[15] W. Pasterkamp and H. Pacejka. OPTIMAL DESIGN OF NEURAL NETWORKS FOR ESTIMATION OF TYRE/ROAD FRICTION. *Vehicle System Dynamics* **29**.sup1 (Jan. 1998), 312–321. ISSN: 0042-3114, 1744-5159. DOI: 10.1080/00423119808969567. URL: http://www.tandfonline.com/doi/abs/10.1080/00423119808969567 (visited on 07/28/2019).

[16] T. Song, H. Zhou, and H. Liu. "Road friction coefficient estimation based on BP neural network". *2017 36th Chinese Control Conference (CCC)*. Dalian, China: IEEE, July 2017, pp. 9491–9496. ISBN: 978-988-15639-3-4. DOI: 10.23919/ChiCC.2017.8028871. URL: http://ieeexplore.ieee.org/document/8028871/ (visited on 07/14/2019).

[17] G. Panahandeh, E. Ek, and N. Mohammadiha. "Road friction estimation for connected vehicles using supervised machine learning". *2017 IEEE Intelligent Vehicles Symposium (IV)*. Los Angeles, CA, USA: IEEE, June 2017, pp. 1262–1267. ISBN: 978-1-5090-4804-5. DOI: 10.1109/IVS.2017.7995885. URL: http://ieeexplore.ieee.org/document/7995885/ (visited on 07/02/2019).

[18] S. Chen. "Road Friction Estimation using Machine Learning". Master's Thesis. Chalmers University of Technology, 2018.

[19] S. Song et al. "Estimating the Maximum Road Friction Coefficient with Uncertainty Using Deep Learning". *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI: IEEE, Nov. 2018, pp. 3156–3161. ISBN: 978-1-72810-321-1 978-1-72810-323-5. DOI: 10.1109/ITSC.2018.8569965. URL: https://ieeexplore.ieee.org/document/8569965/ (visited on 06/28/2019).

[20] Arvi Jonnarth. "Camera-Based Friction Estimation with Deep Convolutional Neural Networks". PhD thesis. Uppsala: Uppsala Universitet, July 2018.

[21] L. Chen et al. Estimation of tire-road friction coefficient based on frequency domain data fusion. *Mechanical Systems and Signal Processing* **85** (Feb. 2017), 177–192. ISSN: 08883270. DOI: 10.1016/j.ymssp.2016.08.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327016302825 (visited on 07/28/2019).

[22] Z. Alyafeai and L. Ghouti. A fully-automated deep learning pipeline for cervical cancer classification. *Expert Systems with Applications* **141** (2020), 112951.

[23] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[24] N. Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* **15** (2014), 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[25] E. Liu. *Model Selection*. 2019. URL: http://ethen8181.github.io/machine-learning/model_selection/model_selection.html (visited on 02/25/2020).

[26] A. M. Ribeiro et al. Estimation of Tire-Road Friction for Autonomous Vehicles: a Neural Network Approach. *arXiv:1908.00452 [cs, eess]* (Aug. 2019). arXiv: 1908.00452. URL: http://arxiv.org/abs/1908.00452 (visited on 08/15/2019).

[27] O. Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* **115**.3 (2015), 211–252. DOI: 10.1007/s11263-015-0816-y.

# A VCC Features

Table A.1: Data collected by VCC's vehicle sensors.

| | **Variable** | **Description** |
|---|---|---|
| 1 | VehMInit | - |
| 2 | AbsMode_B_1 | Anti-Brake system control is active or not for front left wheel |
| 3 | AbsMode_B_2 | Anti-Brake system control is active or not for front right wheel |
| 4 | AbsMode_B_3 | Anti-Brake system control is active or not for rear left wheel |
| 5 | AbsMode_B_4 | Anti-Brake system control is active or not for rear right wheel |
| 6 | AccPedlRat_P | Acceleration pedal ratio |
| 7 | BrkPedlRat_P | Break pedal ratio |
| 8 | CltchPedlRat_P | Clutch pedal ratio |
| 9 | Ax_mps2 | Longitudinal acceleration [m/s2] |
| 10 | Ay_mps2 | Lateral acceleration [m/s2] |
| 11 | Az_mps2 | Vertical acceleration [m/s2] |
| 12 | BrkTqWhl_Nm_1 | Break torque for front left wheel |
| 13 | BrkTqWhl_Nm_2 | Break torque for front right wheel |
| 14 | BrkTqWhl_Nm_3 | Break torque for rear left wheel |
| 15 | BrkTqWhl_Nm_4 | Break torque for rear right wheel |
| 16 | DoorIsOpen_B | Door or trunk is open |
| 17 | PtTqWhl_Nm_1 | Powertrain torque for front axle |
| 18 | PtTqWhl_Nm_2 | Powertrain torque for rear axle |
| 19 | LongCltTq_Nm | Longitudinal clutch status |
| 20 | LongCltSt_M | Longitudinal clutch status |
| 21 | DtSts_M_1 | Front axle is engaged in drive train |
| 22 | DtSts_M_2 | Rear axle is engaged in drive train |
| 23 | StabBrkMode_B_1 | ABS/ESC/RSC intervention for front left wheel |
| 24 | StabBrkMode_B_2 | ABS/ESC/RSC intervention for front right wheel |
| 25 | StabBrkMode_B_3 | ABS/ESC/RSC intervention for rear left wheel |
| 26 | StabBrkMode_B_4 | ABS/ESC/RSC intervention for rear right wheel |
| 27 | StabPtMaxMode_B_1 | Engine traction control for front left wheel is at maximum |
| 28 | StabPtMaxMode_B_2 | Engine traction control for front right wheel is at maximum |
| 29 | StabPtMinMode_B_1 | Engine traction control for rear left wheel is at maximum |
| 30 | StabPtMinMode_B_2 | Engine traction control for rear right wheel is at maximum |
| 31 | WhlAng_r | Steering wheel angle |
| 32 | TrlSt_B | - |
| 33 | WhlSpd_mps_1 | Wheel speed for front left wheel |
| 34 | WhlSpd_mps_2 | Wheel speed for front right wheel |
| 35 | WhlSpd_mps_3 | Wheel speed for rear left wheel |
| 36 | WhlSpd_mps_4 | Wheel speed for rear right wheel |
| 37 | Wx_rps | Roll rate |
| 38 | Wz_rps | Yaw rate |
| 39 | TsaMode_B | - |
| 40 | FrackFr_N | Rack force in steering column |
| 41 | TrsmParkLockd_B | Transmission park is active |
| 42 | EpbSts_M | Electric Parking brake status |
| 43 | WhlSpdDir_M_1 | Wheel speed direction of rear left wheel |
| 44 | WhlSpdDir_M_2 | Wheel speed direction of rear right wheel |
| 45 | AmbTemp_deg | Ambinent temperature in Celsius |
| 46 | Gear | Current active gear |
| 47 | EngN_rpm | Engine rotational speed [rpm] |
| 48 | TorsBarTq_Nm | Steering wheel torque |

| | | |
|---|---|---|
| 49 | VertWhlDsp_m_1 | - |
| 50 | VertWhlDsp_m_2 | - |
| 51 | VertWhlDsp_m_3 | - |
| 52 | VertWhlDsp_m_4 | - |
| 53 | GpsSpd_mps_1 | GPS longitude change rate |
| 54 | GpsSpd_mps_2 | GPS longitude change rate |
| 55 | GpsSpd_mps_3 | GPS longitude change rate |
| 56 | GpsPos_1 | Longitude position |
| 57 | GpsPos_2 | Latitude position |
| 58 | GpsPos_3 | Altitude position |
| 59 | GpsDir_r | GPS heading [rad] |
| 60 | GpsDOP_1 | GPS horizontal dilution of precision |
| 61 | GpsDOP_2 | GPS vertical dilution of precision |
| 62 | GpsDOP_3 | GPS position dilution of precision |
| 63 | GpsDOP_4 | GPS time dilution of precision |
| 64 | GpsNofS | Number of GPS satellites |
| 65 | WhlCogCnt_1 | Wheel cog counters for front left wheel |
| 66 | WhlCogCnt_2 | Wheel cog counters for front right wheel |
| 67 | WhlCogCnt_3 | Wheel cog counters for rear left wheel |
| 68 | WhlCogCnt_4 | Wheel cog counters for rear right wheel |
| 69 | Fx_N_1 | Longitudinal force of front left wheel |
| 70 | Fx_N_2 | Longitudinal force of front right wheel |
| 71 | Fx_N_3 | Longitudinal force of rear left wheel |
| 72 | Fx_N_4 | Longitudinal force of rear right wheel |
| 73 | Fy_N_1 | Lateral force of front left wheel |
| 74 | Fy_N_2 | Lateral force of front right wheel |
| 75 | Fy_N_3 | Lateral force of rear left wheel |
| 76 | Fy_N_4 | Lateral force of rear right wheel |
| 77 | Fz_N_1 | Vertical force of front left wheel |
| 78 | Fz_N_2 | Vertical force of front right wheel |
| 79 | Fz_N_3 | Vertical force of rear left wheel |
| 80 | Fz_N_4 | Vertical force of rear right wheel |
| 81 | EulerVxMode_M | - |
| 82 | WhlSlip_1 | Slip rate of front left wheel |
| 83 | WhlSlip_2 | Slip rate of front right wheel |
| 84 | WhlSlip_3 | Slip rate of rear left wheel |
| 85 | WhlSlip_4 | Slip rate of rear right wheel |
| 86 | Vx_mps | Longitudinal velocity at center of gravity |
| 87 | WhlBpNrj | Bandpass-filtered energy in suspension mode |
| 88 | Friction_1 | Friction on left wheel |
| 89 | Friction_2 | Friction on right wheel |
| 90 | Qly | - |

# B SMHI's road weather features

Table B.1: Data collected by SMHI's weather stations.

| | **Feature** | **Description** |
|---|---|---|
| 1 | Lufttemperatur (h) | Air Temperature, measured every hour |
| 2 | Lufttemperatur (dygn) | Average Air Temperature over a day |
| 3 | Lufttemperatur (månad) | Average Air Temperature over a month |
| 4 | Lufttemperatur, min och max (12h) | Min & Max Air Temperature, over a 12h period |
| 5 | Lufttemperatur, min och max (dygn) | Min & Max Air Temperature, over a day |
| 6 | Daggpunktstemperatur (h) | Dew Point Temperature, measured every hour |
| 7 | Nederbördsmängd (15 min) | Amount of precipitation over a 15 min period |
| 8 | Nederbördsmängd (h) | Amount of precipitation over a 1 hour period |
| 9 | Nederbördsmängd (dygn) | Amount of precipitation over a 1 day period |
| 10 | Nederbördsmängd (månad) | Amount of precipitation over a 1 month period |
| 11 | Nederbördsintensitet (15 min) | Maximum precipitation intensity measured over a 15 min period |
| 12 | Nederbördstyp (12h) | Observed precipitation types over a 12 hour period |
| 13 | Nederbördstyp (dygn) | Observed precipitation types over a 24 hour period |
| 14 | Snödjup (dygn) | Snow depth, measured once per day |
| 15 | Relativ luftfuktighet (h) | Relative Air Humidity, measured every hour |
| 16 | Vindriktning och vindhastighet (h) | Wind direction and wind speed. Average over a 10 min period, once every hour. |
| 17 | Vindhastighet, max av medel (h) | Maximum of the average wind speed (10 min) over the past 3 hours. |
| 18 | Byvind, max (h) | Maximum wind speed measured over 2 seconds, during a period of 1 hour |
| 19 | Total molnmängd (h) | Total amount of clouds, measured once every hour |
| 20 | Signifikanta moln (h) | Air Temperature, measured every hour |
| 21 | Lägsta molnbas (h) | Distance from the ground to the "lowest" cloud, measured once every hour. |
| 22 | Lägsta molnbas, min (15 min) | Shortest distance from the ground to the "lowest" cloud, during a 15 min period. |
| 23 | Solskenstid (h) | Amount of sun hours accumulated during a period of 1 hour |
| 24 | Globalstrålning (h) | Average natural radiation, measured over a period of 1 hour. |
| 25 | Långvägsstrålning (h) | Average long-wave radiation, measured over a period of 1 hour |
| 26 | Lufttryck (h) | Air Pressure compared to sea-level, measured every hour |
| 27 | Sikt (h) | Sight / Viewing distance, measured every hour |
| 28 | Rådande väder (h) | Current weather, measured every hour |

# C Trafikverket's road weather features

Table C.1: Data collected by Trafikverket's road sensor stations.

| | **Feature** | **Description** |
|---|---|---|
| 1 | TLuft °C | Air Temperature |
| 2 | Daggp °C | Dew Point Temperature |
| 3 | TYta °C | Ground Temperature, measured 2mm above the surface. |
| 4 | TYta - Daggp °C | Dew Point Ground Temperature, measured 2mm above the surface. |
| 5 | Snö mm | Snow Volume |
| 6 | Regn mm | Rain Volume |
| 7 | Smält mm | Melted snow volume |
| 8 | Lufu % | Humidity |
| 9 | Vind m/s | Average Wind Speed over the last 30 min |
| 10 | Vindmax m/s | Maximum Wind Speed during the last 30 min |
| 11 | Virik ° | Current Wind Direction |
| 12 | Nedbtyp | Precipitation type |

# D Weather Feature Selection - the impact of individual weather features on the model performance

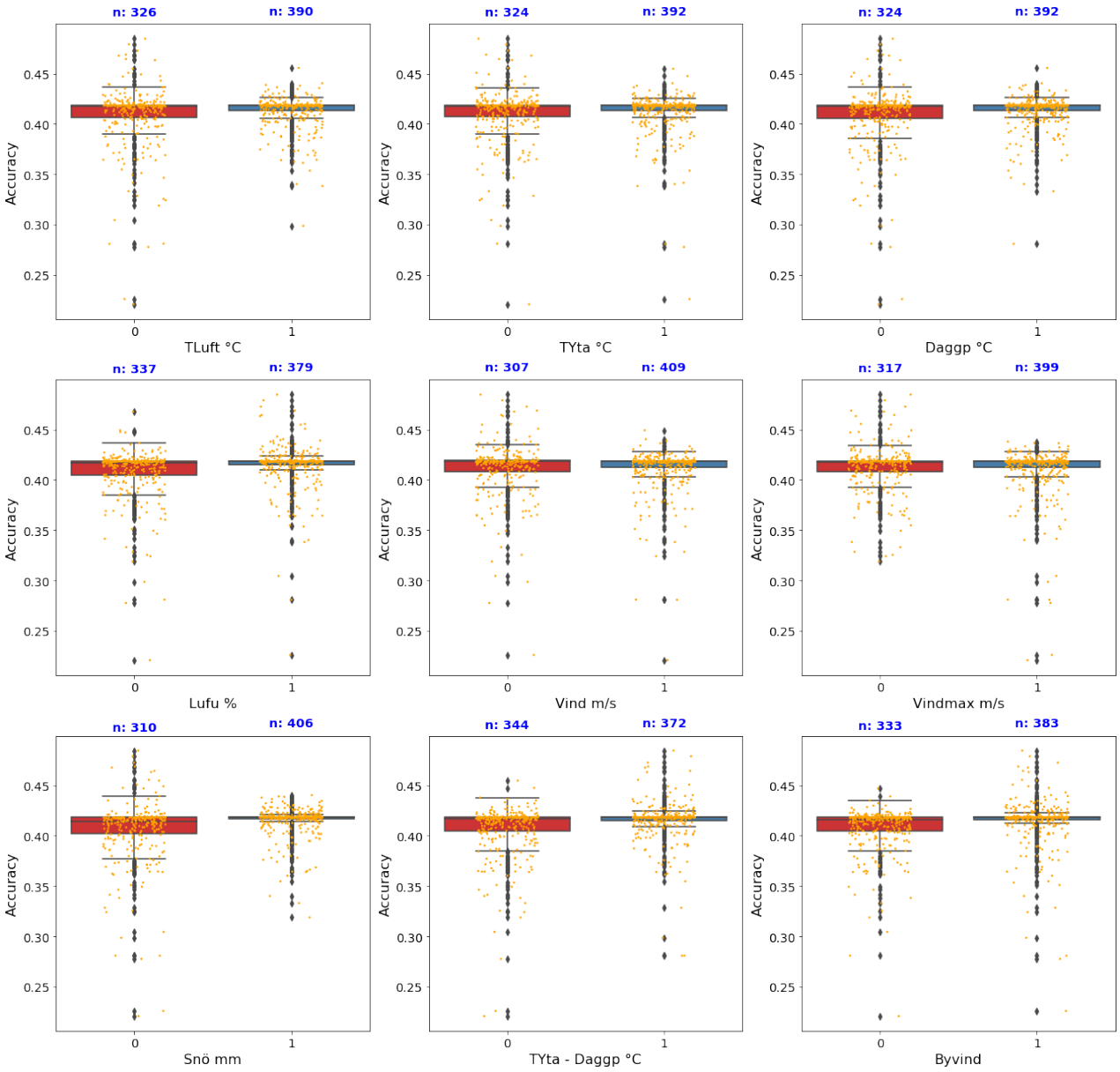



Figure D.1: Boxplots showing the accuracy distribution of our model (y-axis) depending on the presence (1) or non-presence (0) of the various weather features (x-axis). Above each boxplot is a number representing the amount of tests that were performed with each specific setting. This was part of the final process for selecting the most suitable weather features.

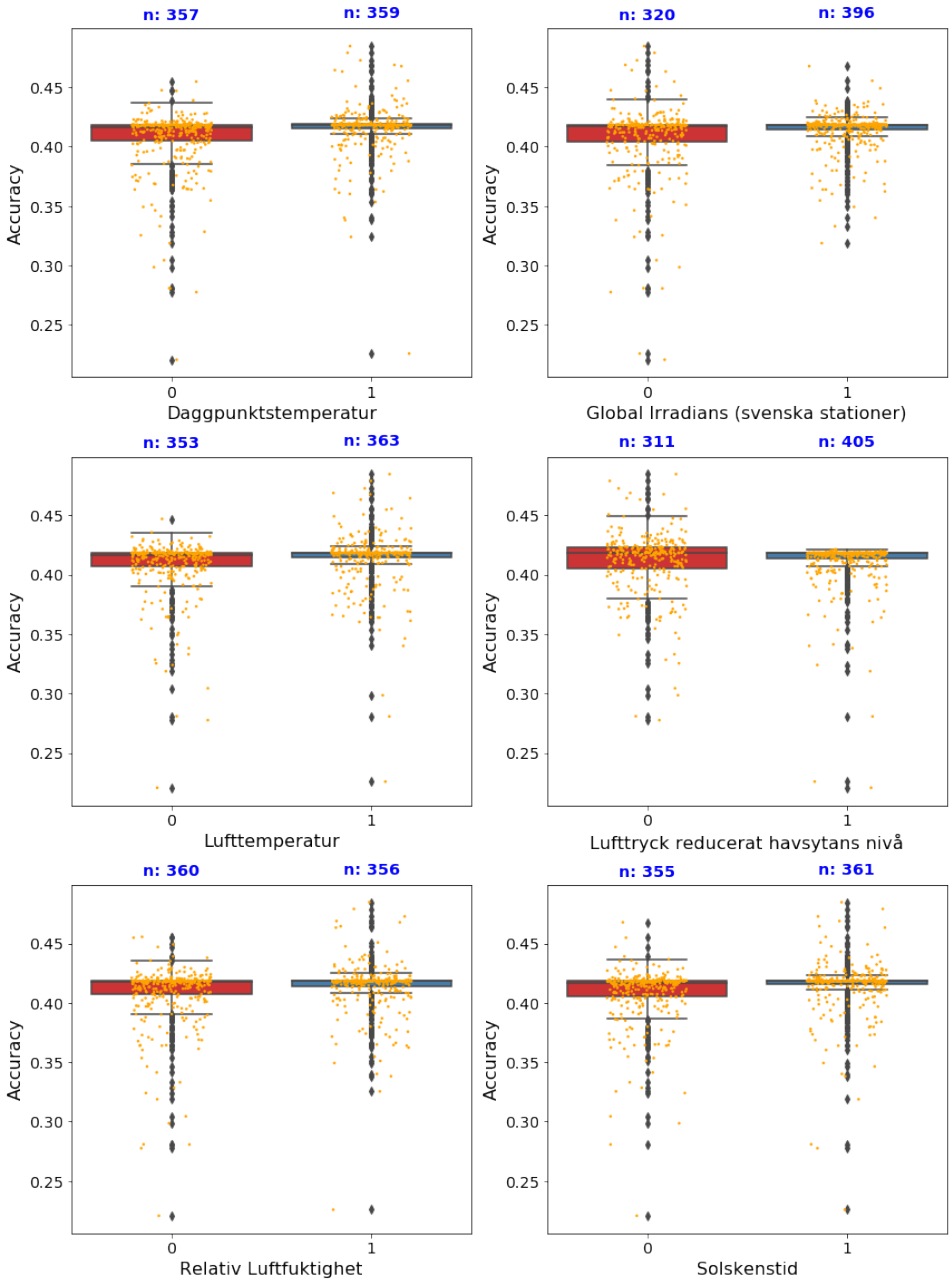



Figure D.2: Boxplots showing the accuracy distribution of our model (y-axis) depending on the presence (1) or non-presence (0) of the various weather features (x-axis). Above each boxplot is a number representing the amount of tests that were performed with each specific setting. This was part of the final process for selecting the most suitable weather features.

# E Confusion Matrix Results

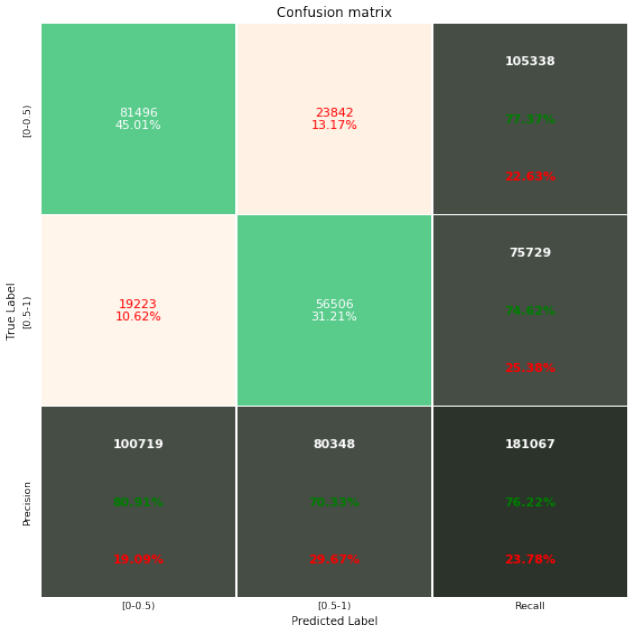



Figure E.1: Confusion Matrix of the binary classification task, showing a model's predictions compared to the true underlying labels.

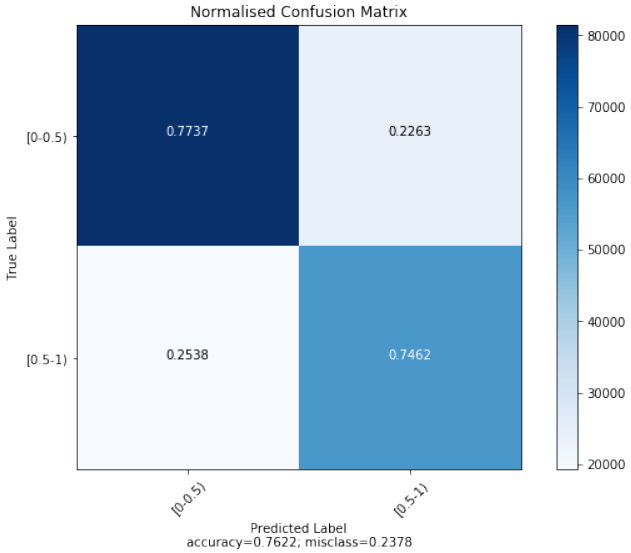



Figure E.2: Confusion Matrix of the binary classification task, showing a model's *normalised* predictions compared to the true underlying labels.

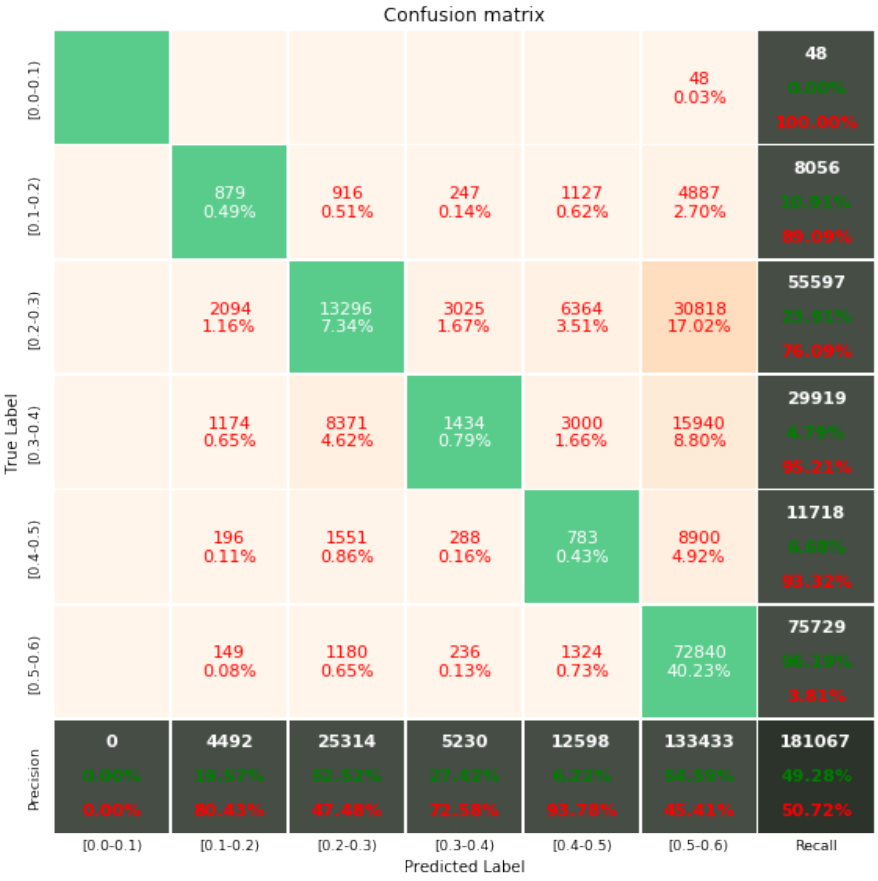



Figure E.3: Confusion Matrix of the 6-class classification task, showing a model's predictions compared to the true underlying labels.

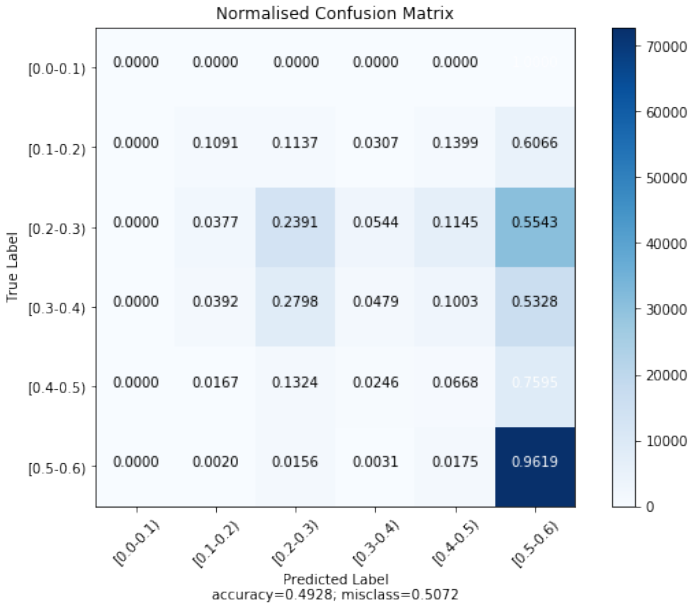



Figure E.4: Confusion Matrix of the 6-class classification task, showing a model's *normalised* predictions compared to the true underlying labels.

# F VCC Data Distribution

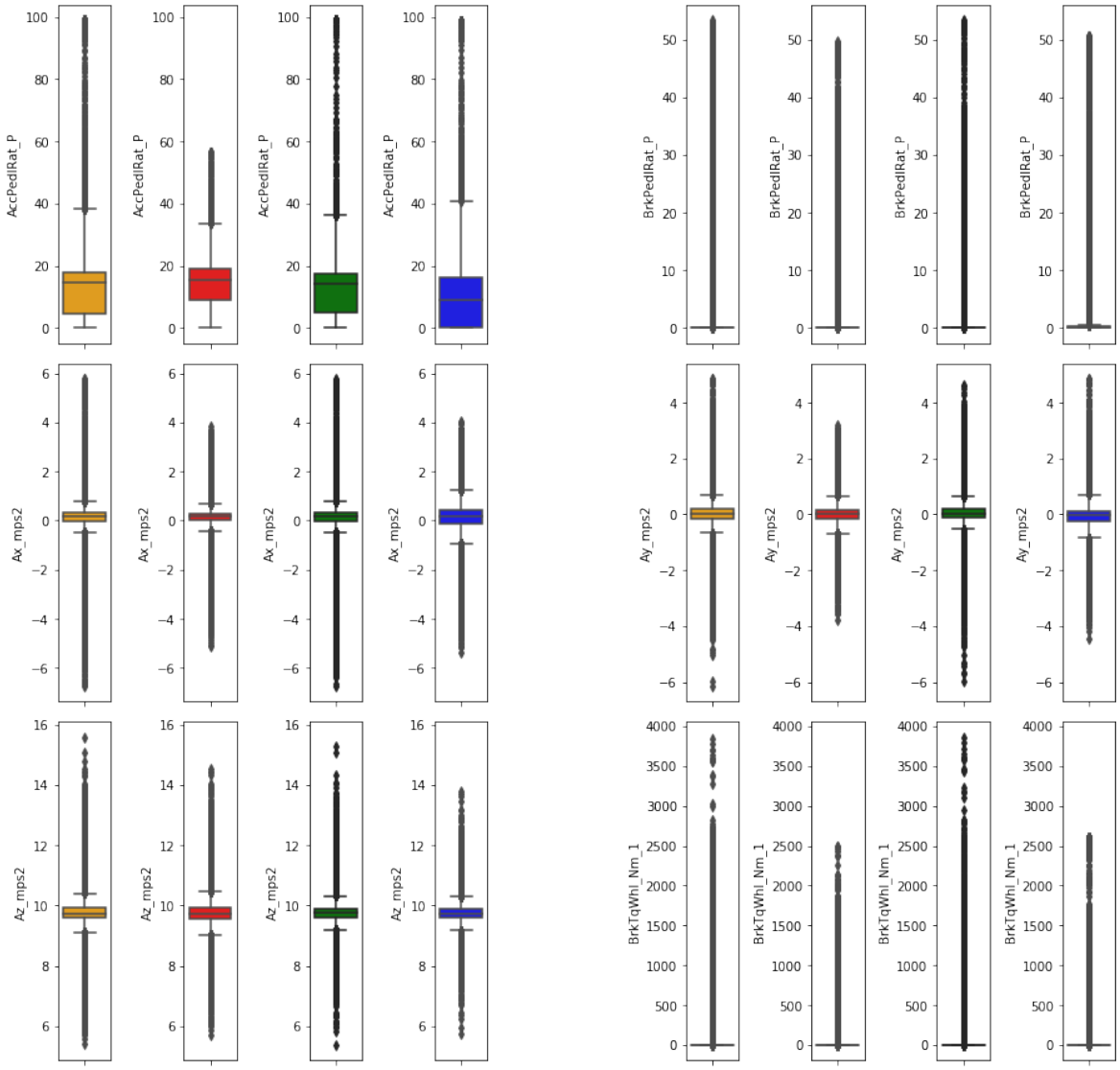



Figure F.1: VCC Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every VCC feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

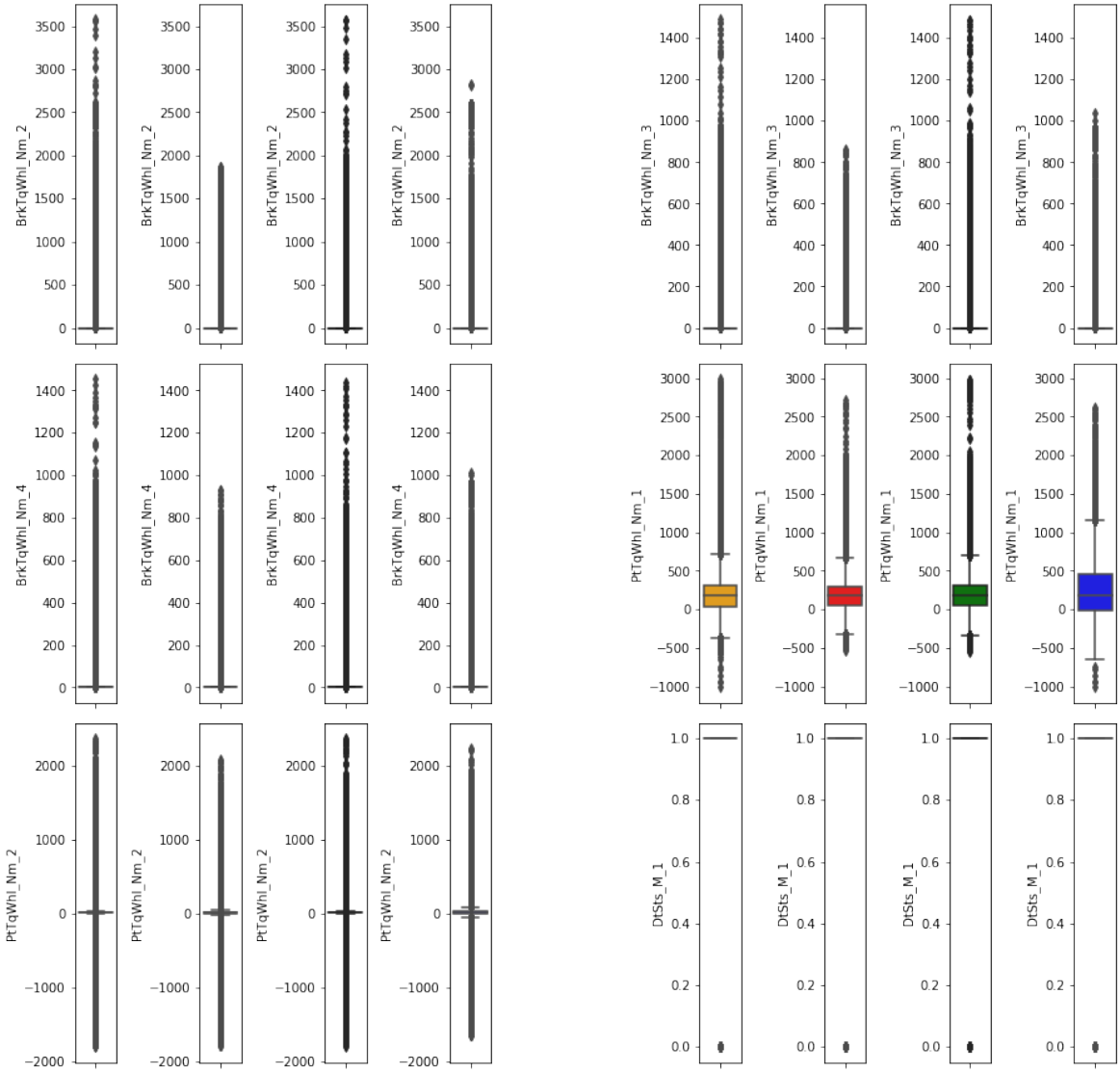



Figure F.2: VCC Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every VCC feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

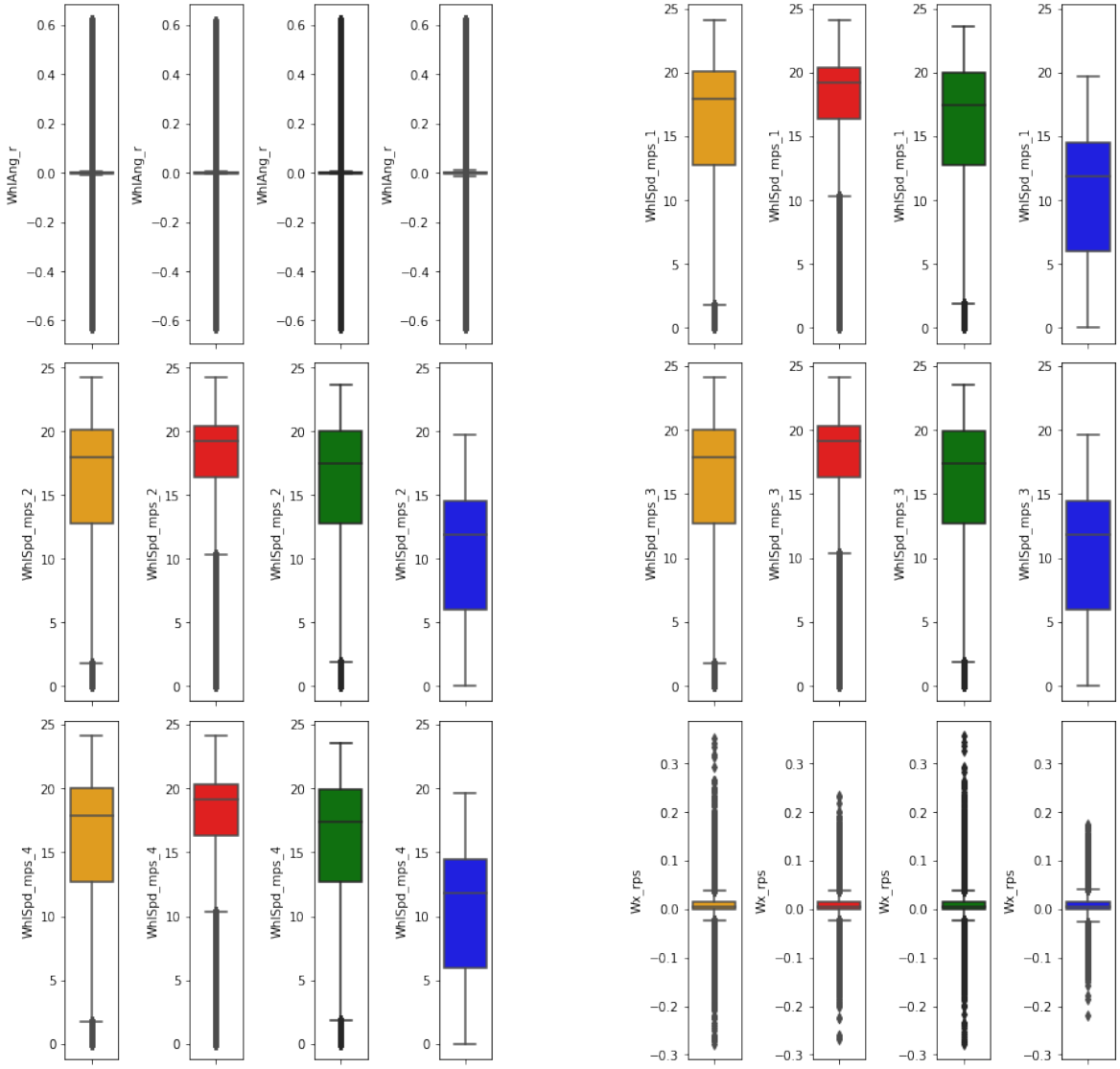



Figure F.3: VCC Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every VCC feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

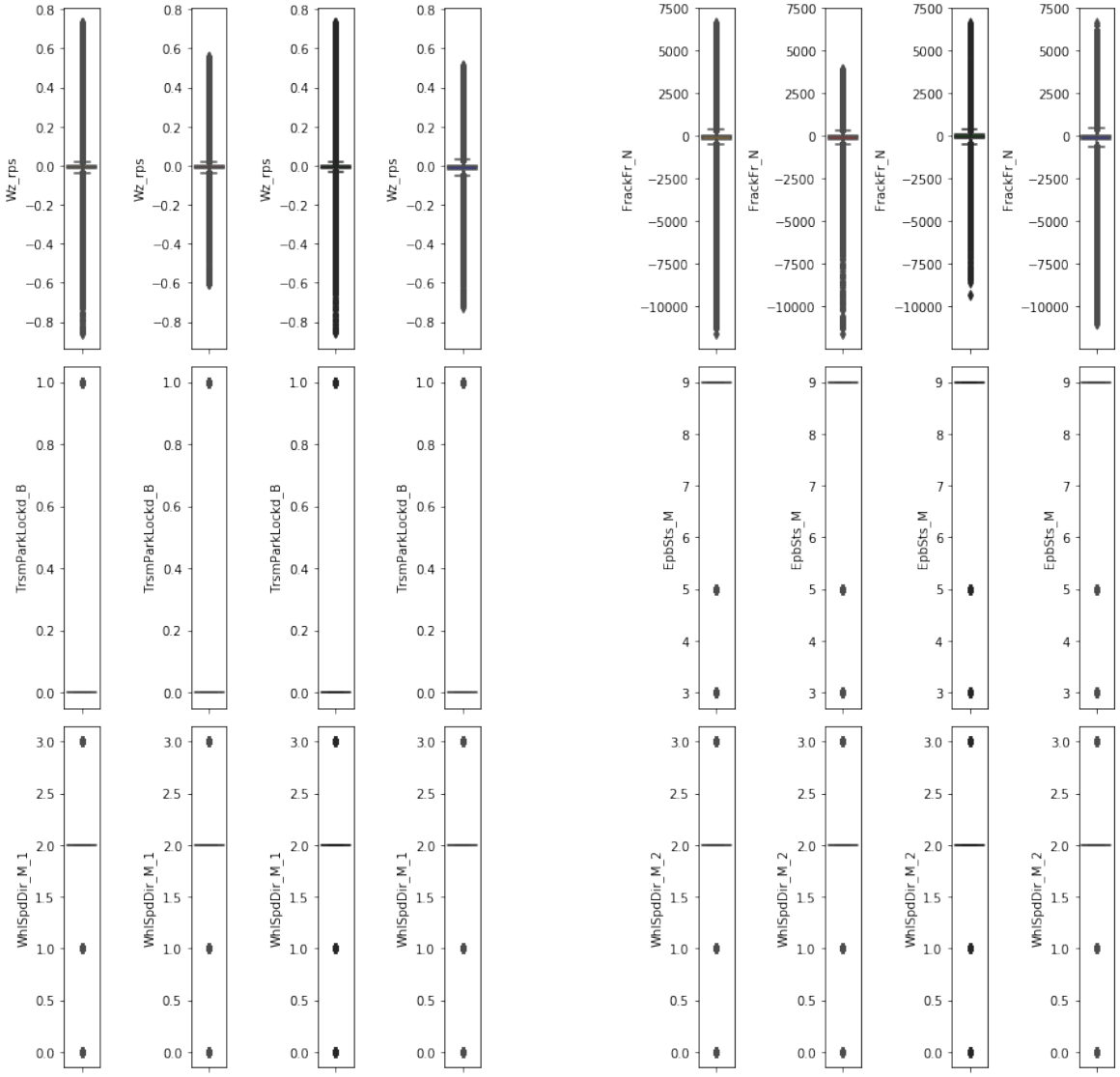



Figure F.4: VCC Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every VCC feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

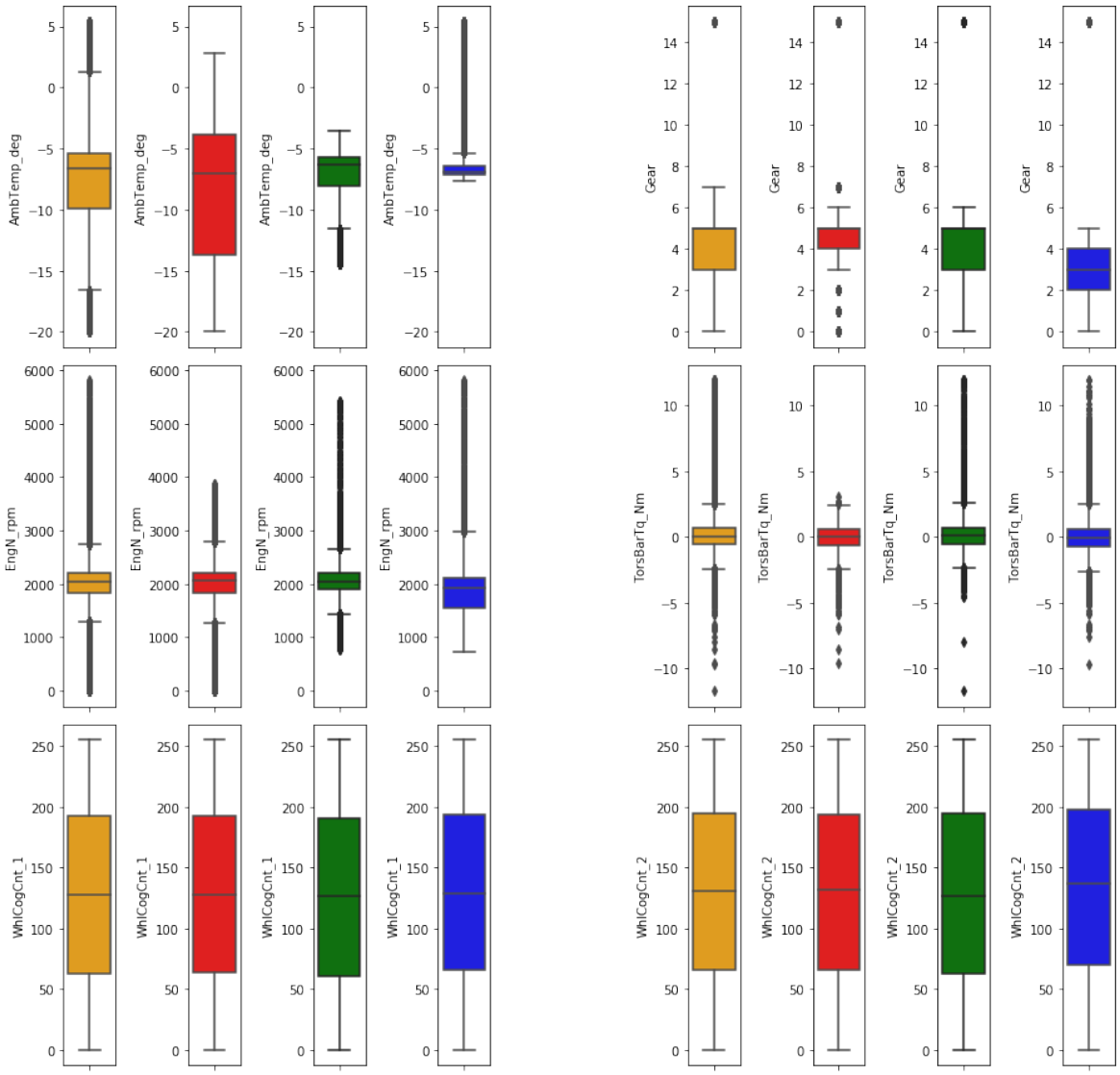



Figure F.5: VCC Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every VCC feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

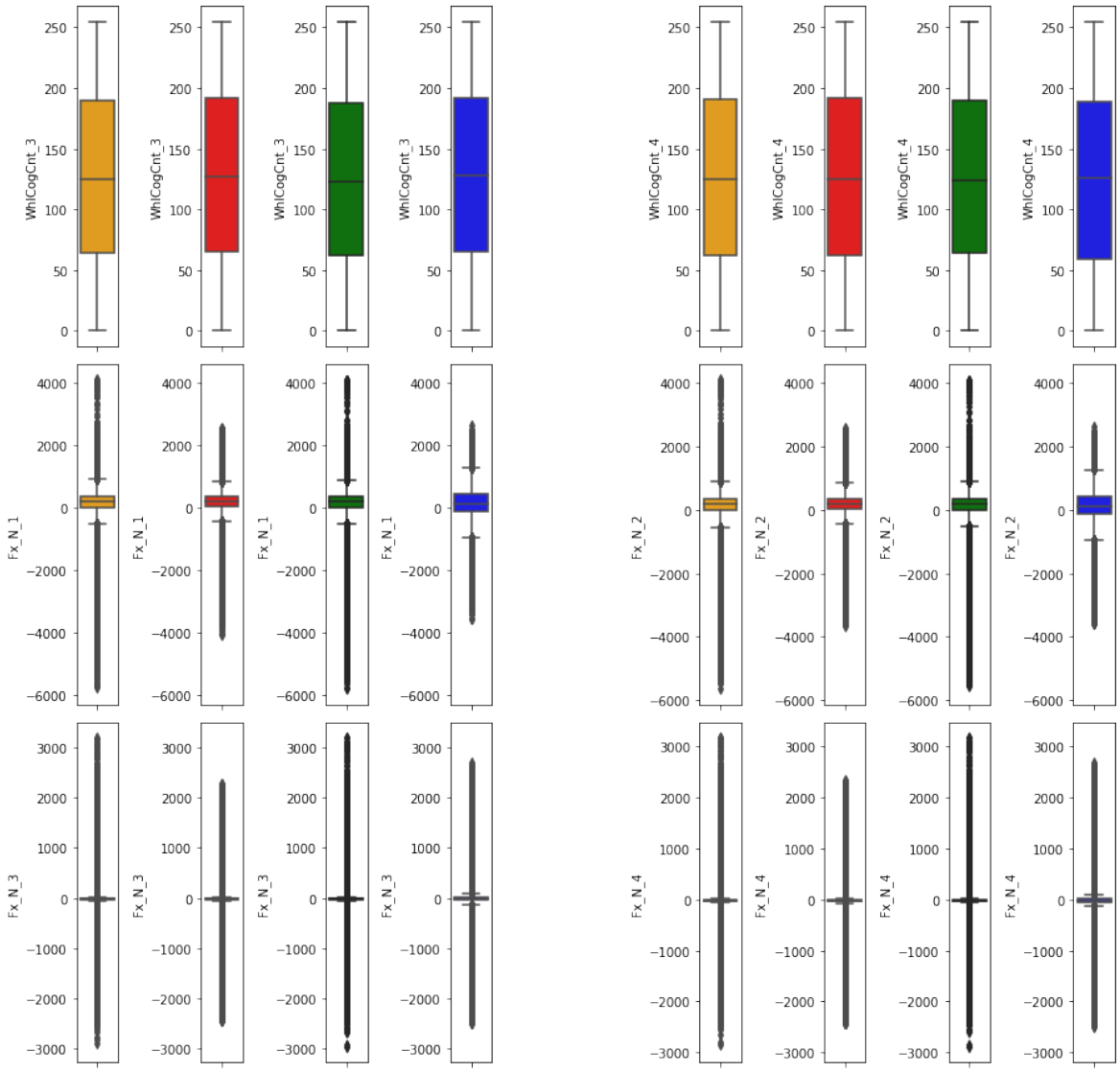



Figure F.6: VCC Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every VCC feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

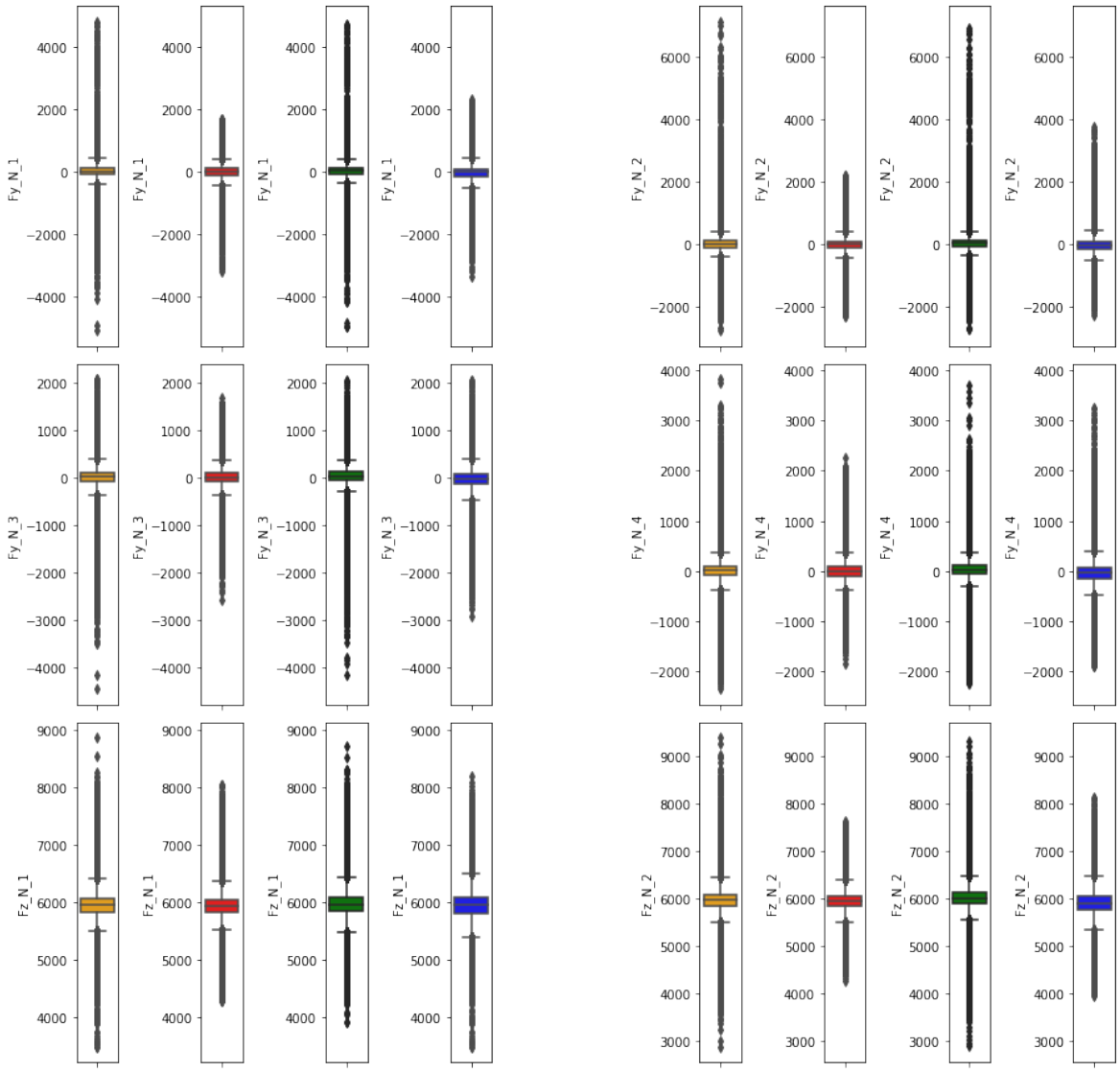



Figure F.7: VCC Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every VCC feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

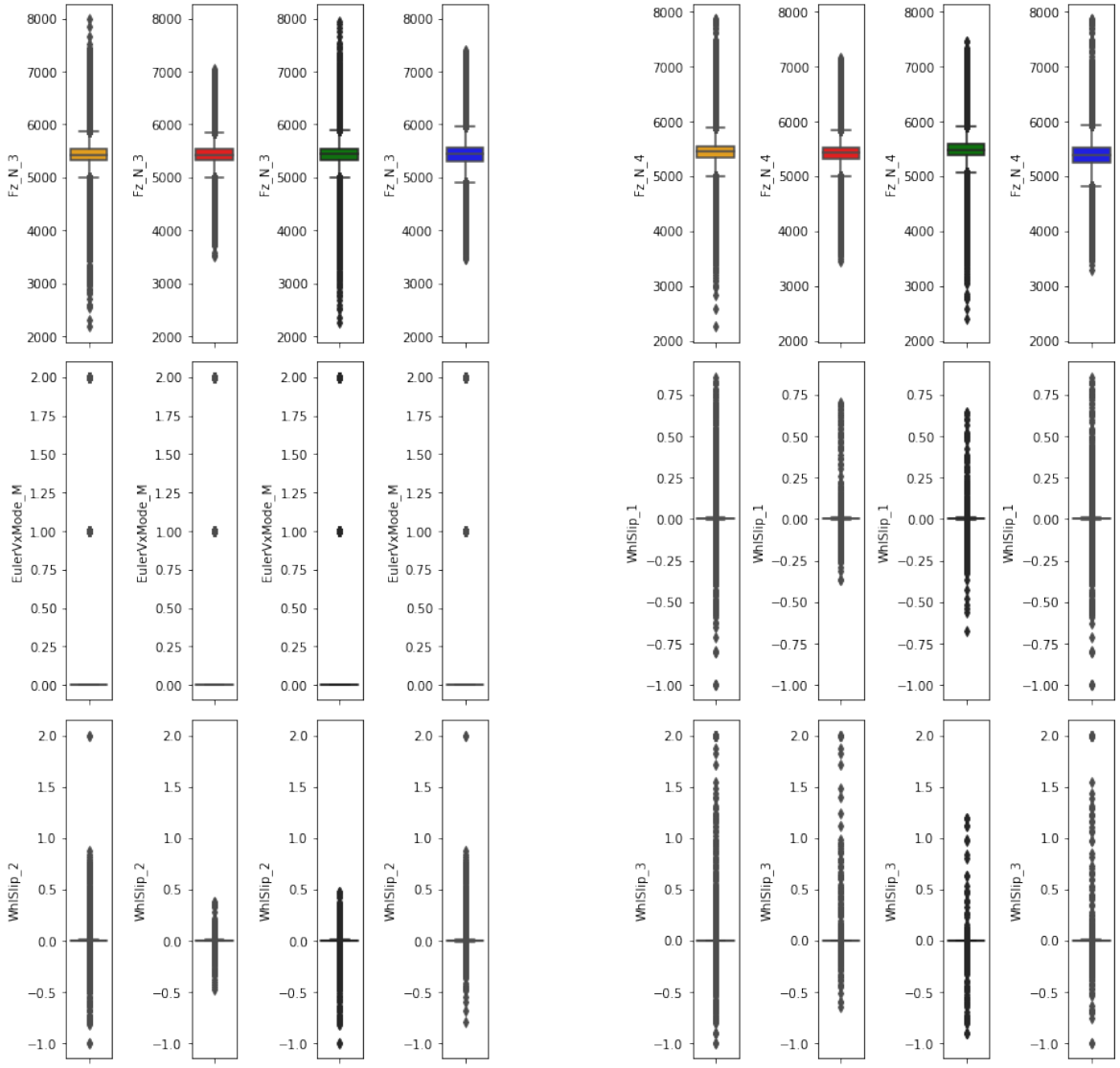



Figure F.8: VCC Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every VCC feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

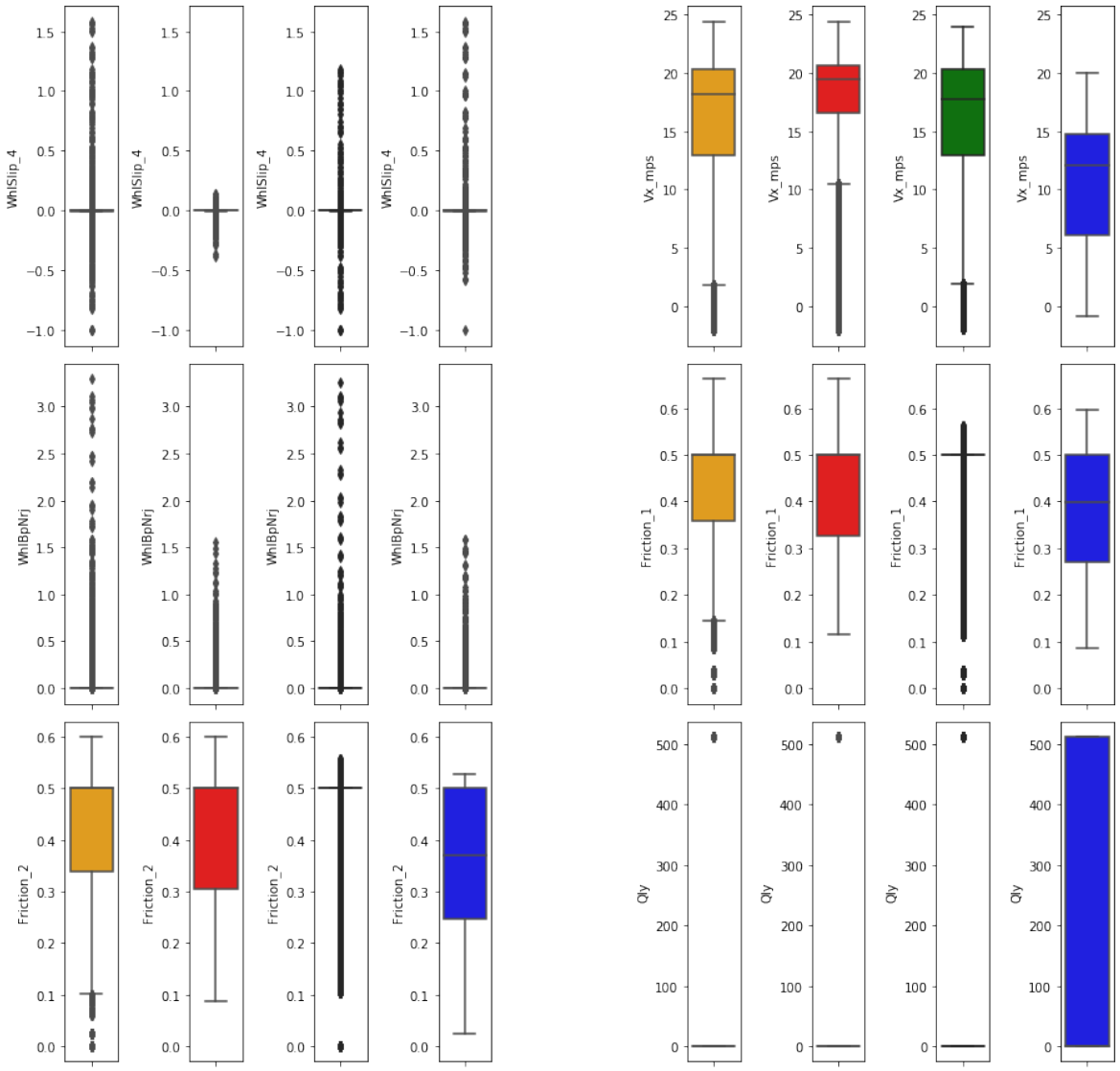



Figure F.9: VCC Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every VCC feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

# G Weather Data Distribution

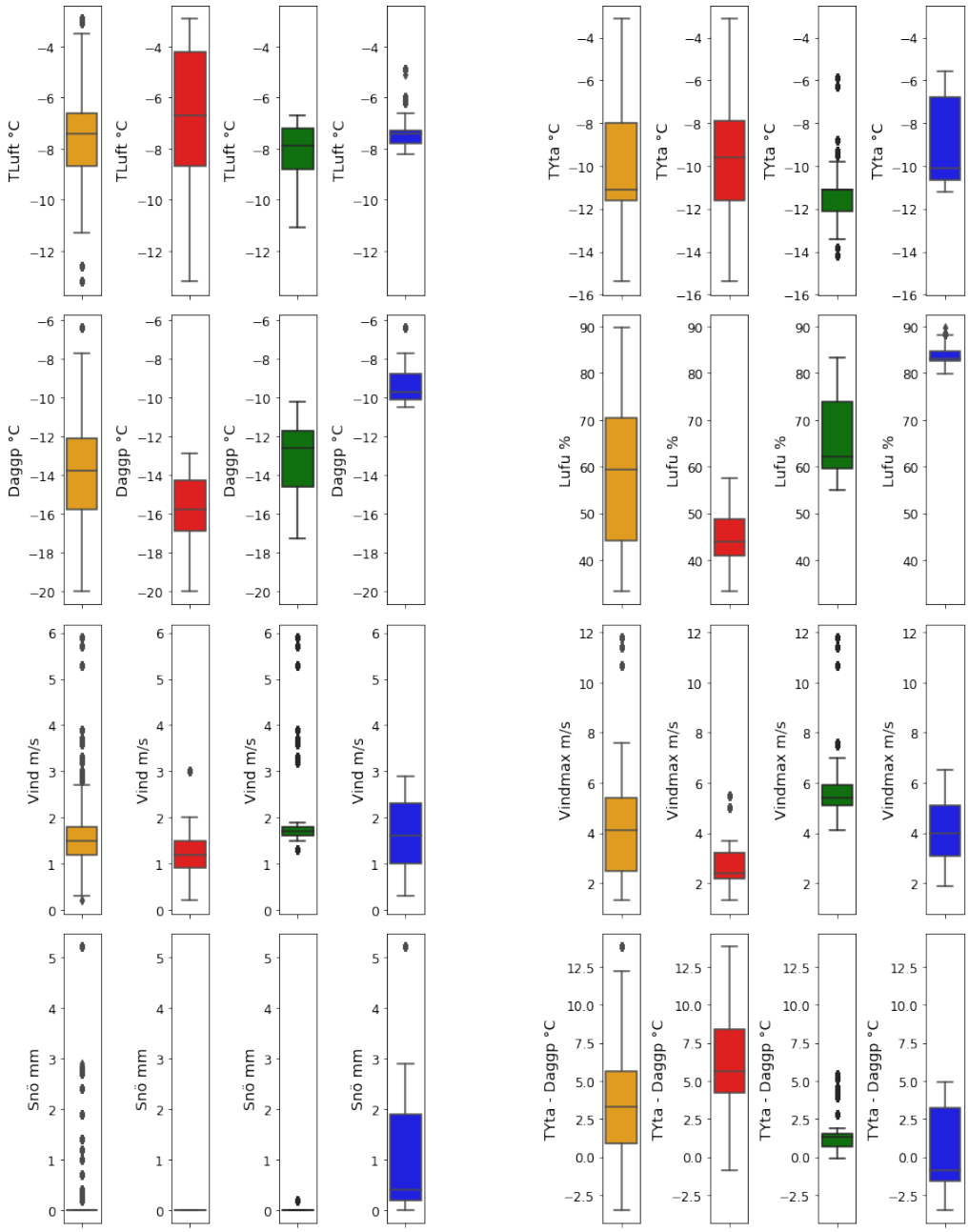



Figure G.1: Weather Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every weather feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

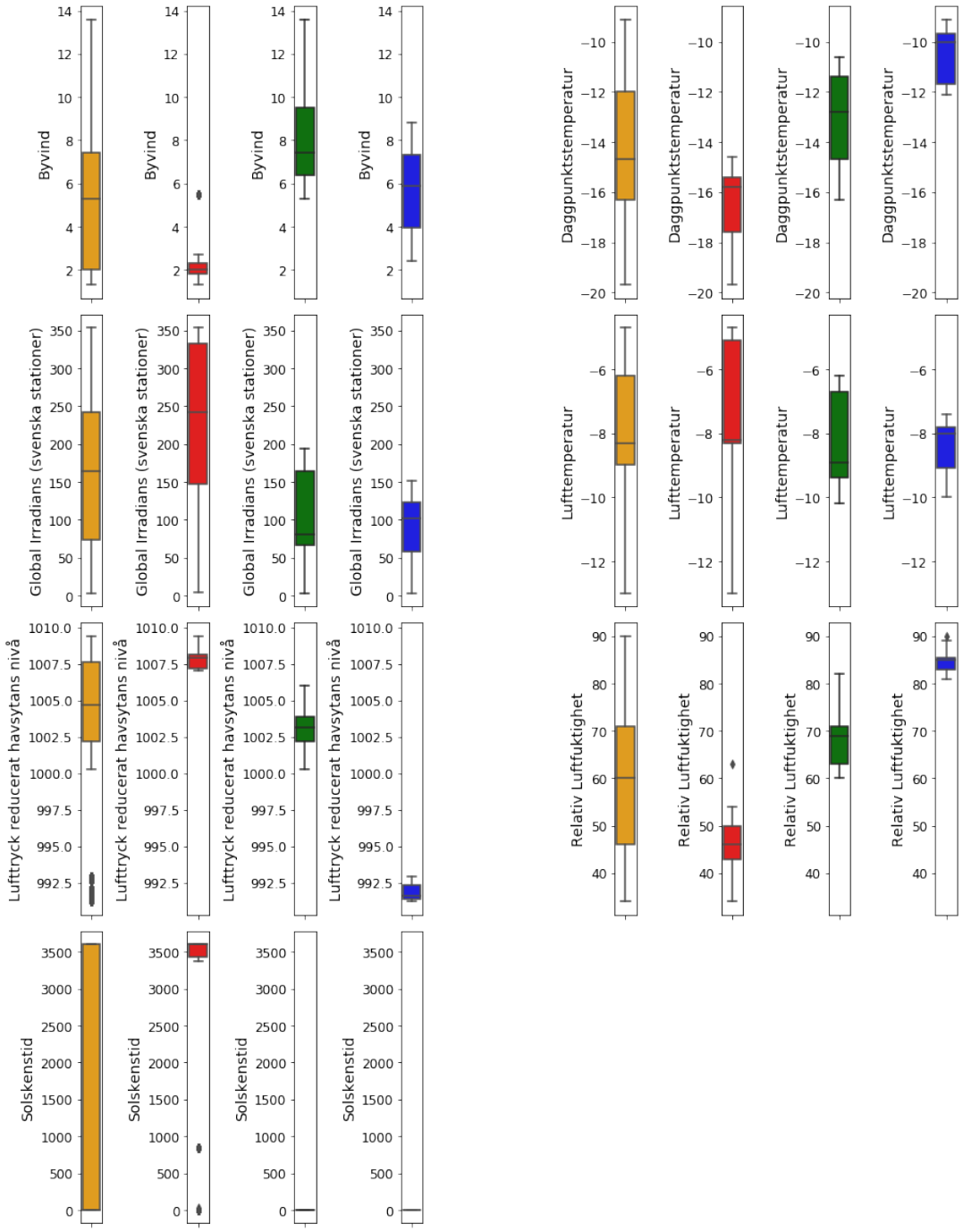



Figure G.2: Weather Data Feature Distributions, modelled as a boxplot. There are 4 boxplots for every weather feature; the first one (orange) is the aggregated distribution over all days, the second one (red) is for the data from day 2, the third one (green) is for the data from day 3 and the fourth one (blue) is for the data from day 4 (the test set).

# H Data Feature Correlation Analysis

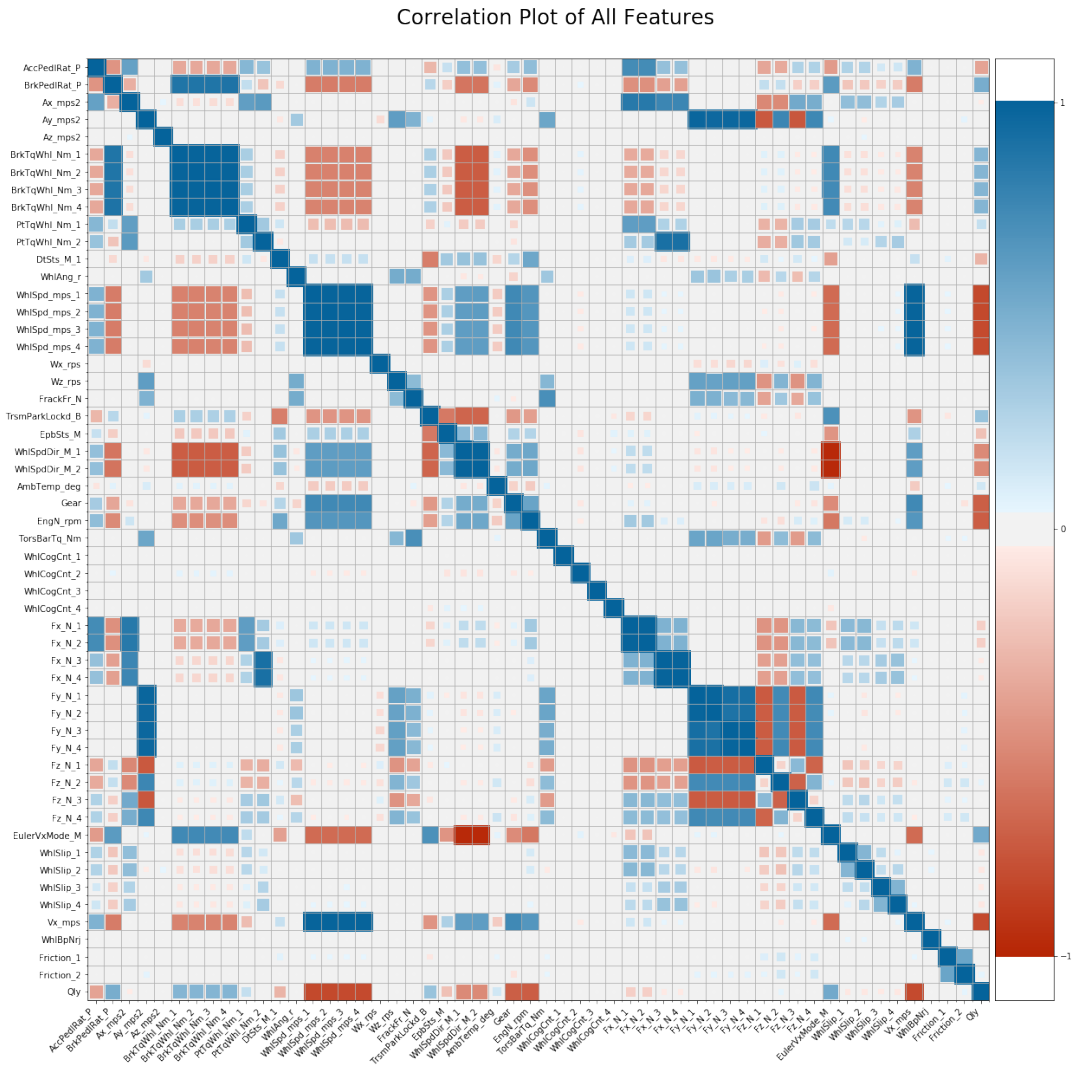



Figure H.1: Correlation analysis of all VCC features, minus the features that have been deemed unlikely to contribute to our model in a meaningful way. A cell with bigger and deeper blue color signifies a stronger positive relation, while a cell with bigger and deeper red color signifies a stronger negative relation.

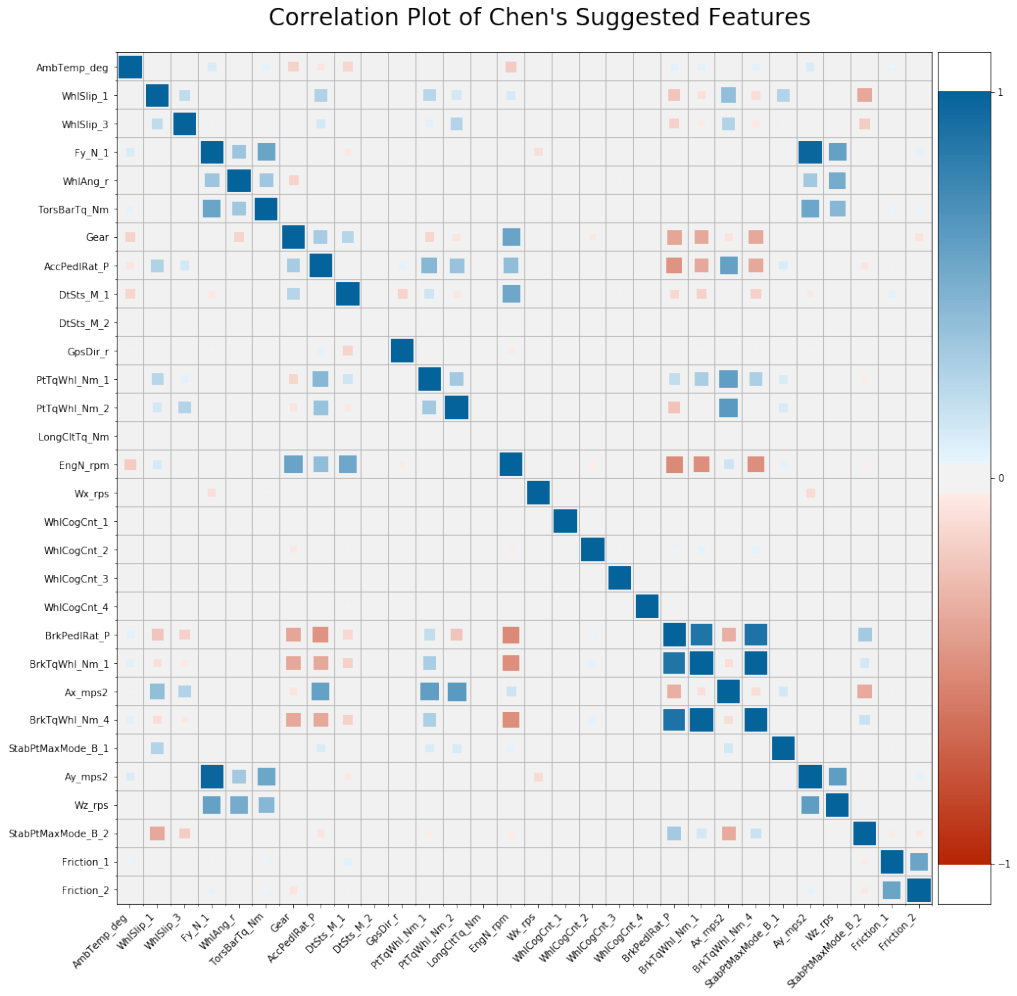



Figure H.2: Correlation analysis of the VCC features suggested by Chen [18]. A cell with bigger and deeper blue color signifies a stronger positive relation, while a cell with bigger and deeper red color signifies a stronger negative relation.