Acknowledgements

This thesis marks the final step for me to achieve a master's degree in computer science. Six years have soon passed since I started studying at Luleå university of Technology, a time in my life where I have learned a lot and met wonderful people. I would like to thank classmates, friends and teachers which I have got to know during my years in Luleå for making this a memorable and educative journey.

This thesis would not be possible without guidance from my supervisors. Firstly I would like to thank Niklas Karvonen at LTU for the work of reviewing my thesis, providing advice on how to write academically, and for all the machine learning teachings which proved valuable for a machine learning newcomer like myself. Secondly I would like to thank Johan Casselgren at LTU for giving me the opportunity of carrying out this thesis with Trafikverket, and for all the help I have received throughout the process. I would also like to thank Jonas Hallenberg at Trafikverket for providing a dataset which I could work with, and for answering any questions I had regarding Road Weather Information Systems throughout this project.

Luleå, June 2018

Tobias Axelsson

Abstract

Trafikverket, the agency in charge of state road maintenance in Sweden, have a number of so-called Road Weather Information Systems (RWIS). The main purpose of the stations is to provide winter road maintenance workers with information to decide when roads need to be plowed and/or salted. Each RWIS have a number of sensors which make road weather-related measurements every 30 minutes. One of the sensors is dug into the road which can cause traffic disturbances and be costly for Trafikverket. Other RWIS sensors fail occasionally.

This project aims at modelling a set of RWIS sensors using supervised machine learning algorithms. The sensors that are of interest to model are: Optic Eye, Track Ice Road Sensor (TIRS) and DST111. Optic Eye measures precipitation type and precipitation amount. Both TIRS and DST111 measure road surface temperature. The difference between TIRS and DST111 is that the former is dug into the road, and DST111 measures road surface temperature from a distance via infrared laser. Any supervised learning algorithm trained to model a given measurement made by a sensor, may only train on measurements made by the other sensors as input features. Measurements made by TIRS may not be used as input in modelling other sensors, since it is desired to see if TIRS can be removed. The following input features may also be used for training: road friction, road surface condition and timestamp.

Scikit-learn was used as machine learning software in this project. An experimental approach was chosen to achieve the project results: A pre-determined set of supervised algorithms were compared using different amount of top relevant input features and different hyperparameter settings. Prior to achieving the results, a data preparation process was conducted. Observations with suspected or definitive errors were removed in this process. During the data preparation process, the timestamp feature was transformed into two new features: month and hour.

The results in this project show that precipitation type was best modelled using Classification And Regression Tree (CART) on Scikit-learn default settings, achieving a performance score of $\overline{F1}_{test} = 0.46$ and accuracy = 0.84 using road surface condition, road friction, DST111 road surface temperature, hour and month as input features. Precipitation amount was best modelled using k-Nearest Neighbor (kNN); with $k = 64$ and road friction used as the only input feature, a performance score of $MSE_{test} = 0.31$ was attained. TIRS road surface temperature was best modelled with Multi-Layer Perceptron (MLP) using 64 hidden nodes and DST111 road surface temperature, road surface condition, road friction, month, hour and precipitation type as input features, with which a performance score of $MSE_{test} = 0.88$ was achieved. DST111 road surface temperature was best modelled using Random forest on Scikit-learn default settings with road surface condition, road friction, month, precipitation type and hour as input features, achieving a performance score of $MSE_{test} = 10.16$.

Sammanfattning

Trafikverket är den organisation som ansvarar för underhåll av statliga vägar i Sverige. Till hjälp för att utföra detta har de ett antal så kallade Vägväderinformationssystem (VVIS) som framförallt syftar till att underlätta arbetet för de som arbetar med vintervägunderhåll. Informationen används för att bestämma om vägar behöver skottas och/eller saltas. Varje VVIS har ett antal sensorer som gör väder-relaterade mätningar var trettionde minut. En av dessa sensorer grävs ned i marken, vilket kan störa trafikflöden och införa kostnader för Trafikverket. Övriga sensorer fallerar relativt ofta, vilket kan ställa till problem för vintervägunderhållsarbetarna om de behöver information just då.

Det här projektet syftar till att modellera ett antal VVIS sensorer med hjälp av övervakade maskininlärningsalgoritmer. Följande sensorer är intressanta att modellera: Optic Eye, Track Ice Road Sensor (TIRS) och DST111. Optic Eye mäter nederbördstyp och nederbördsmängd. Både TIRS och DST111 mäter väglagstemperatur, skillnaden är att TIRS är nedgrävd i vägbanan, medan DST111 mäter på avstånd via infraröd laser. En sensor som modelleras av en övervakad maskininlärningsalgoritm, får träna på data från övriga sensorer, förutom från TIRS då det är intressant att se om denna kan tas bort. Dessutom får följande input features användas för träning: vägfriktion, väglagstyp och tidsstämpel.

Scikit-learn användes som maskininlärningsprogram i det här projektet. En experimentell strategi användes för att uppnå projektets resultat: Ett förbestämt antal algoritmer jämfördes där antalet mest relevanta input features och hyperparametrar varierades. En dataförberedelseprocess genomfördes innan resultaten nåddes. Observationer med antingen misstänkta eller definitiva fel togs bort från datasetet i detta steg. Under dataförberedelseprocessen transformerades även tidsstämpel som feature till två nya features: månad och timma.

Resultaten i detta projektet visar att nederbördstyp modellerades bäst genom att använda Classification And Regression Tree (CART) på standardinställningar i Scikit-learn. En prestanda på $\overline{F1}_{test} = 0.46$ och noggrannhet = 0.84 uppnåddes när väglagstyp, vägfriktion, DST111 väglagstemperatur, timma och månad användes som input features. Nederbördsmängd modellerades bäst genom att använda k-Nearest Neighbor (kNN) med $k = 64$ och vägfriktion som input feature. Med detta uppnåddes en prestanda på $MSE_{test} = 0.31$. TIRS väglagstemperatur modellerades bäst genom att använda Multi-Layer Perceptron (MLP) med 64 gömda noder och DST111 väglagstemperatur, väglagstyp, vägfriktion, månad, timma och nederbördstyp som input features då en prestanda på $MSE_{test} = 0.88$ åstadkoms. DST111 modellerades bäst genom att använda Random forest på standardinställningar i Scikit-learn och med väglagstyp, vägfriktion, månad, nederbördstyp och timma som input features. Med detta nåddes en prestanda på $MSE_{test} = 10.16$.

Contents

Chapter 1

Introduction

The chapter starts with a background describing who Trafikverket are, what Road Weather Information Systems are, and why the technology behind them needs improvement. This is followed up with explaining machine learning basics and how machine learning can be used in this project. An objective for the project is defined followed by its delimitations. The chapter ends with an overview of the provided dataset and the thesis structure.

1.1 Background

Living in cold areas of the world typically means an increased amount of work for invididual people, municipalities and companies in trying to maintain a non-winter-like infrastructure. This of course, also involves winter road maintenance. Salting and plowing roads is an investment in not only saving lives, but also in lowering socio-economic costs; Arvidsson [1] presents two scenarios which explains this claim: The two scenarios take place on a road with 2 cm snow and a daily traffic flow of 2000 vehicles, one with a salted and ploughed road taking four hours to drive, and the other scenario on the same road without winter maintenance taking five hours to drive. Arvidson argues that the total socio-economic costs are 3.5% higher in the non-maintained road, mainly due to increased travel time and thus, higher accident costs.

1.1.1 Road Weather Information Systems

While the socio-economic savings in performing winter road maintenance may be enough to justify the need for it, winter road maintenance can still present a notable economic cost for the organization(s) involved. Trafikverket, the agency in charge of state road maintenance in Sweden, reported that winter road maintenance were roughly 18% of the total road maintenance costs in 2013 [2]. Local contracters are hired to carry out the plowing and salting of state roads, with requirements on both ends regarding when to plow, which roads to prioritize etc. [3]. Trafikverket help contracters monitor road conditions with their so-called Road Weather Information Systems (RWIS) [3]. Trafikverket have approximately 800 RWIS distributed across state roads in Sweden [3].

Figure 1.1 – RWIS station at sensor site Myggsjön [4].

Table 1.1 – Measurements that are studied in this project from RWIS with corresponding instrument- or sensor names[5], [6].

Instrument/sensor name	Feature	Value	Measured at how many RWIS
Optic Eye	precipitation type	discrete	all
Optic Eye	precipitation amount	continous	all
Track Ice Road Sensor	road surface temperature	continous	all
DST111	road surface temperature	continous	~ 7
DSC111	road surface condition	discrete	~ 26
DSC111	road friction	continous	~ 26
MS4	timestamp	date (mmddhhmm)	all

Table 1.1 shows some of the sensors that the operational RWIS are equipped with. In addition to the sensors listed in table 1.1, many of the stations are also equipped with a camera [7]. The operational RWIS also measure air humidity, air temperature, max-wind, wind-average and more. The sensors are connected to a nearby computer called MS4. Trafikverket aim to replace MS4 with a new generation of computers by 2021 [8]. MS4 has limited capacity to handle current and future contracter needs, and electric components are becoming hard for Trafikverket to replace [8].

Over personal communication with Johan Casselgren at Luleå University of Technology, Johan mentions that it is interesting to investigate if certain sensors, especially the Track Ice Road Sensor (TIRS), can be replaced along with the new generation of computers. TIRS is required to be dug into the road, which may require roads to be closed off temporarily during installation. Furthermore, if the sensor is removed, a hole is left in the road which can cause problems. In that way, the DST111 may prove more useful since it measures road temperature remotely using infrared laser [9].

The author, Johan Casselgren and Niklas Karvonen, who is also from Luleå University of Technology, hypothesized over personal communication that machine learning models can probably be used to model the behavior of TIRS. The author and Johan Casselgren also sees potential in using machine learning models as a backup system when sensors malfunction. It is therefore interesting to see if the other sensors from table 1.1 can be modelled as well.

1.1.2 Machine learning

Machine learning as formally defined by Mitchell [10]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E". This means that machine learning algorithms are used to solve a set of problems, measure its performance in doing so, and ultimately improve in some way from previous experiences. For example, imagine a program designed to determine if a human face is in a photo or not. Since photos are taken at different distances, angles and faces have different characteristics such as eye color, skin color, distance between eyes and nose shape, implementing this "manually" may prove cumbersome. Instead of programming an algorithm to recognize faces, it can be programmed *to learn to recognize faces*. If the algorithm is allowed to analyze a dataset with thousands of photos of human faces, it could learn to distinguish a human face by recognizing parts of the face such as eyes, nose, mouth and where those parts are most likely placed to oneanother.

In essence, machine learning algorithms improve/learn in some way from processing a dataset. How they learn can be used to broadly categorize machine learning algorithms as either having supervised or unsupervised learning [11]. Supervised learning algorithms process a labeled dataset while unsupervised learning attempts to make sense of unlabeled data. The data provided by Trafikverket to perform this project is labeled data in the form of column headers in Microsoft Excel workbooks (see provided data in 1.4).

Dimensionality in machine learning refers to how many features are used as input to train a machine learning algorithm. A tempting brute-force approach to a machine learning problem may be to include every possible feature available to build a model, not only those brought up in table 1.1, but also road photos, wind-speed etc. This however, may introduce the curse of dimensionality, which basically means that higher dimensional data introduces complexity and possibly an increased amount of errors in the model [12]. During a personal meeting, Johan Casselgren recommended the author to focus on using the features listed in table 1.1. However, according to Jonas Hallenberg who works at Trafikverket, there are difficulties in trying to model the DSC111 sensor. The problem is that Trafikverket use salt on every road where the DSC111 sensor is, save one, and salt affects the road surface status. So for example, when the surface temperature and dew point temperature suggests that road surface condition is icy, which could be measured by DSC111, it may be wet in reality. Johan casselgren suggests that data from the DSC111 sensor can still be used as input when modelling the other sensors. Data from TIRS road temperature is not to be used as input when modelling other sensors since

Trafikverket may remove this sensor in the future.

The forementioned definition of machine learning by [10] mentions a performance measure P. A performance measure can be used to evaluate a supervised learning algorithm's abilitiy to model a given feature. A specific feature that is to be modelled is referred to as target feature, and the features a supervised learning algorithm uses to do so is referred to as input features. In a basic sense, a supervised learning algorithm studies the values of the target feature, and attempts to build a model that best fits its behavior based on data from input features. But the algorithm is not necessarily perfect by having a high performance score on its training dataset. If a model is built in such a way that it fits its provided training data perfectly, which may have statistical outliers, errors etc., it may have difficulties in predicting unseen data. This condition is known as overfitting, the opposite: underfitting, both of which are covered in detail in section 2.3. This means it is not enough for a supervised learning algorithm to have a high performance, it should also generalize well on unseen data.

1.2 Objective

The objective of this thesis is to find optimized supervised learning algorithms, in terms of performance and generalization, which model the behavior of the following sensors: Optic Eye, TIRS and DST111. Each sensor is modelled using measurements made by the other sensors, except for TIRS. Any algorithm may train on the following input features as well: timestamp, road friction and road surface condition. Timestamp is provided by the RWIS computer MS4. Road friction and road surface condition are measured by a sensor called DSC111.

The objective is broken down to four subtasks that are to be solved:

1. Find the algorithm among a set of supervised learning algorithms, that can be used to classify **precipitation type**, with the best performance- and generalization score. The algorithm may use the following input features:
    - timestamp
    - DST111 road surface temperature
    - road friction
    - road surface condition
2. Find the algorithm among a set of supervised learning algorithms, that can be used to predict **precipitation amount**, with the best performance- and generalization score. The algorithm may use the following input features:
    - timestamp
    - DST111 road surface temperature
    - road friction
    - road surface condition

3. Find the algorithm among a set of supervised learning algorithms, that can be used to predict **TIRS road surface temperature**, with the best performance- and generalization score. The algorithm may use the following input features:

    - timestamp
    - precipitation type
    - precipitation amount
    - DST111 road surface temperature
    - road friction
    - road surface condition

4. Find the algorithm among a set of supervised learning algorithms, that can be used to predict **DST111 road surface temperature**, with the best performance- and generalization score. The algorithm may use the following input features:

    - timestamp
    - precipitation type
    - precipitation amount
    - road friction
    - road surface condition

1.3 Delimitations

There are many supervised learning algorithms, all of which are not evaluated in detail in this project. An algorithm is qualified for evaluation in this project if the following is true:

1. The algorithm is a supervised learning algorithm that solves regression and/or multiclass classification problems (see classification and regression in 2.1.1 and 2.1.2). This criteria ensures that the algorithm can be used to solve at least one of the project subtasks.

2. The algorithm is available in Scikit-learn, which is the machine learning software library used in this project.

3. The algorithm belongs to one of the following algorithm families (see information on the different algorithm families in 2.2):

    - Decision tree based learning
    - Instance based learning
    - Bayesian learning
    - Regression based learning

- Artificial Neural Networks
- Ensemble learning

The purpose of choosing algorithms from the families listed above is that the families are assumed to capture the majority of supervised learning algorithms, based on similarity.

The performance of supervised algorithms can generally be improved by optimizing its hyperparameters (see information on hyperparameters in 2.4). However, there can be many ways that a single algorithm can be configured. This project deals with comparing performances of several algorithms, and to explore all combinations of hyperparameter settings is too time-consuming. When it comes to optimizing hyperparameters, it was decided to focus on the choice of k in kNN, the number of hidden nodes in MLP and magnitude of regularization λ in Lasso (see details in 2.2 and 2.3.2).

Some of the sensors, such as the DSC111, are malfunctioning frequently (see the amount of errors reported by DSC111 in figure 4.1). To avoid any complex relationships between functioning and malfunctioning sensors, it was decided to only model non-error behavior by using non-error observations.

In addition to the timestamp feature, an extra feature was present in the provided dataset which displayed what year each observation was taken. The author assumes that global warming does not present a significant change in weather conditions, road surface temperature etc. over a period of two years. Furthermore, since the measurements are from 2015-2016, it is assumed by the author that any model built using year as an input feature could potentially do well when dealing with new observations whose year is 2015 or 2016, but that it generalizes poorly when dealing with observations from 2018 or other years. By these assumptions, it was decided to not consider year as an input feature.

Support vector machine (SVM) algorithms could also have been evaluated in this project, but was ruled out since they appear to have high computational running-time. It was revealed by the author, through a cross-validation classification spot-checking experiment (see experimental setups in 3.5.1) on the iris dataset in Scikit-learn, that SVM algorithms have significantly longer running-time than the other algorithms from table 3.1.

1.4 Provided data

A dataset containing 171425 measurements (observations) was provided by Jonas Hallenberg at Trafikverket to carry out this project. The observations are from 2015-2016 from six RWIS along state road E6 in Sweden. The stations measure the features seen in table 1.1, making one measurement every 30 minutes. Six Microsoft Excel workbooks represent data from each of the six stations, in which the column headers are the features seen in table 1.1. The year each observation was taken is also represented in a column in the workbooks.

A readme.txt file was also provided. It explains in words how to interpret the observations. As can be seen in table 1.1, precipitation type and road surface condition have

discrete values that are explained in the readme file. Table 1.2 shows what each value corresponds to for the two sensors.

Table 1.2 – Values that Optic Eye precipitation type and DSC111 road surface condition can assume and what they mean.

Value	Precipitation type	Road surface condition
-9	missing sensor/error	-
0	-	error
1	no precipitation	dry
2	rain with $>= 0°C$ air temperature	moist
3	rain with $< 0°C$ air temperature	wet
4	snow	-
5	-	frost
6	rain and snow mixed	snow
7	-	ice
9	unknown type	slush

1.5 Thesis structure

The thesis is structured in the following way:

- Literature review: In-depth information on supervised learning, supervised learning algorithms, generalization, hyperparameter optimization and data preparation.
- Method: Cover strategies and methods used to solve the project subtasks.
- Data preparation process: A process where the dataset is prepared in order to achieve the best possible results.
- Results and analysis: Solutions to the project subtasks are found and analyzed.
- Summary: Gives a summary of what the project is about, and what results were found.
- Discussion and conclusions: Discussion on the results obtained and the methods used, followed up with conclusions and recommendations.

Chapter 2

Literature Review

This chapter starts by explaining supervised learning and supervised learning algorithms. This is followed up with generalization and hyperparameter optimization. The chapter ends with explaining how data preparation can be used to improve performances of supervised learning algorithms.

2.1 Supervised learning

In supervised learning, the algorithm receives a dataset of labeled observations which are used to build a model used to predict correct values for unseen data. A database table storing weather-related data could for example have thousands of database records (observations) where data in each record belong to certain database column headers (features) such as wind speed w_s, wind direction w_d and time t. The goal of supervised learning is to build a model

$$y = f_{map}(x) \tag{2.1}$$

such that when new input data x_{new} is used, f_{map} can predict y_{new}. The model is built from a dataset which is typically split into three parts:

- Training dataset: Used to fit the model.
- Validation dataset: Used to give an unbiased evaluation of a model built from the training dataset which can potentially be used to update its parameters in order to improve performance [12].
- Test dataset: Gives an unbiased evaluation of the final model.

This approach of splitting the dataset is referred to as holdout throughout this project. According to [12], the proportions of the split is usually 60% training, 30% test and 10% validation while [13] suggests that a common approach is 50% training, 30% validation and 20% test. Success has also been shown by using 90% of the data as training data [14]. It is suggested by [13] to employ the holdout approach in any machine learning

project. However, when using the built-in functionality for holdout with Scikit-learn, the user is not able to specify a validation set [15].

Supervised learning can be thought of as having a teacher supervising the learning process. The correct answers are in the training data and the algorithm learns from being corrected by the teacher. Going back to the forementioned example of the weather station to give a brief example of how a supervised machine learning algorithm builds a model: Suppose a training and test dataset is provided, and one wishes to predict wind speed $y = w_s$ based on wind direction and time $x = [x_1, x_2] = [w_d, t]$. During the training process, a supervised learning algorithm goes through the training dataset to build a model, which attempts to minimize its error in predicting wind speed. Suppose the supervised learning algorithm used is Ordinary Least Squares (OLS) (see details on OLS in 2.2.4) and a model is built from the training process:

$$w_s = f_{map}([w_d, t]) = \beta_0 + \beta_1 w_d + \beta_2 t = 4 + 0.2w_d + 1.7t \tag{2.2}$$

The model can then be evaluated using the test dataset to see how it performs on unseen data. Ordinary Least Squares attempts to build a model on a known form, which essentially means it makes assumptions on the dataset beforehand. Algorithms that make such assumptions are known as parametric algorithms. There are also non-parametric algorithms which are more flexible in creating a model. Classification And Regression Tree (CART) is an example of a non-parametric algorithm (see details on CART in 2.2.1).

Estimation of continous output variables, such as wind speed in the example presented above, is a regression problem. In supervised learning there are also algorithms associated with the problem of classification, which deals with categorizing data.

2.1.1 Classification predictive modeling

In a classification problem, the computer is asked to place an observation into one of k categories (classes), where $k \geq 2$ [11]. The problem of classifying previously unseen email as spam or not spam is an example of a classification problem. Google claims that their machine learning models can detect spam and phishing messages with 99.9% accuracy in their widely used Gmail application [16]. Classification on two classes, as in the forementioned example, is known as binary classification, and problems with three or more classes to be classified is called multiclass classification [17]. Classifying precipation type, which is done in this project, is a multiclass classification problem since it has more than two classes.

How classification is done depends on the algorithm used to build the model. These kind of algorithms are commonly known as classifiers. There are several classifiers that can be used to classify observations in a given dataset, but their performance in doing so may differ. Performance of classifiers are typically measured in terms of accuracy, which is the number of correct classifications divided by the number of observations in the test dataset.

$$\text{accuracy} = \frac{\text{n.o. correct classifications test dataset}}{\text{n.o. observations test dataset}} \tag{2.3}$$

A high accuracy score such as 90% may seem promising, but what if 90% of the test data is made up of one class C alone? That means that the model correctly classified all occurrences of C, but failed to classify any of the other classes. This is probably an indication of an imbalanced dataset, which means that the different classes in the dataset are not equally represented. In the case of an imbalanced dataset, [18] claims that accuracy is not a good metric for evaluating classifiers, and that other metrics such as precision-recall break-even, area under the curve etc. should be used instead. None of the performance metrics mentioned by [18] is available in Sckit-learn, but there is a performance metric available known as macro-average $F1$, denoted $\overline{F1}$ throughout this project, that is suitable for evaluating imbalanced multiclass classification problems [19]. It averages the performance of classifying each individual class, in terms of precision- and recall score. Some terminology is presented to simplify the explanation of the different performance metrics:

- True positive (TP): Correctly identified
- False positive (FP): Incorrectly identified
- True negative (TN): Correctly rejected
- False negative (FN): Incorrectly rejected

Equations 2.4, 2.5 and 2.6 shows how recall, precision and $F1$ scores are calculated for a multiclass classification problem on one if its classes A.

$$\text{Recall}(A) = \frac{TP(A)}{TP(A) + FN(A)} \tag{2.4}$$

$$\text{Precision}(A) = \frac{TP(A)}{TP(A) + FP(A)} \tag{2.5}$$

$$F1(A) = 2 \cdot \frac{\text{Recall}(A) \cdot \text{Precision}(A)}{\text{Recall}(A) + \text{Precision}(A)} \tag{2.6}$$

An example to demonstrate how recall and precision works: Suppose a machine learning algorithm is built to diagnose patients with tuberculosis. Suppose the test dataset consists of 100 patients in total, 20 of which have tuberculosis. Say, for example, the software registered 12 patients with tuberculosis, and 10 out of those 12 patients actually had tuberculosis. The precision of the software is $\frac{10}{12}$ and recall is $\frac{10}{20}$.

2.1.2 Regression predictive modeling

In contrast to classification problems, such as classifying incoming email as spam or not spam, regression problems are about predicting continous quantaties. The model presented in equation 2.2 is an example of a regression problem since the goal is to predict a numerical value for wind speed. The problem could be translated into a classification problem by, for example, stating that for given numerical intervals, the

wind speed is categorized as being low, medium or high. This kind of conversion is known as discretization. But even if conversions prove useful, it is beyond the scope of this project.

Performance of a regression model can be measured by computing its mean squared error (MSE) on the test dataset.

$$MSE_{test} = \frac{1}{n}\sum_{i}^{n}(y'_{test} - y_{test})_i^2 \tag{2.7}$$

where y'_{test_i} are predictions on the test and y_{test_i} are actual values. It's a measurement of how close each prediction was to its corresponding target value on average. Although other measurements can be used to evaluate regression models, such as R squared, [20] writes that the primary goal of any regression model evaluation should be to minimize MSE.

2.2 Supervised learning algorithms

There are many algorithms that can be used to solve regression and classification problems, some of which can solve both. There is a famous theorem called the "no free lunch" theorem which contradicts the idea that a single machine learning algorithm is best at solving all possible machine learning problems [21]. It is therefore interesting to study the effects of different supervised learning algorithms in this project, but to explain the functionality of them in detail is out of scope. Instead, the reader is encouraged to look up details on specific algorithms when needed. The algorithms are grouped by similarity in this section, which is referred to as algorithm families. The algorithm families named in the following sections are either taken directly, or as inspiration from [12]. Each family listed in the project delimitations section (see delimitations in 1.3) are covered.

2.2.1 Decision tree based learning

Decision tree algorithms use a decision tree datastructure. Non-leaf nodes represent conditions for a specific feature, and leaves are values of the target feature. One of the main advantages of decision tree algorithms is that they are easy to understand [22]. Figure 2.1 depicts an example of a decision tree used to solve a classification problem with two features: sex and age. Starting at the root node, a decision is made once a leaf-node is reached. Ideally, the feature that best divides the dataset would be represented in the root of the tree, followed by the second best in the second level and so on. However, constructing such optimal decision trees has been proved to be a NP-complete problem [23].

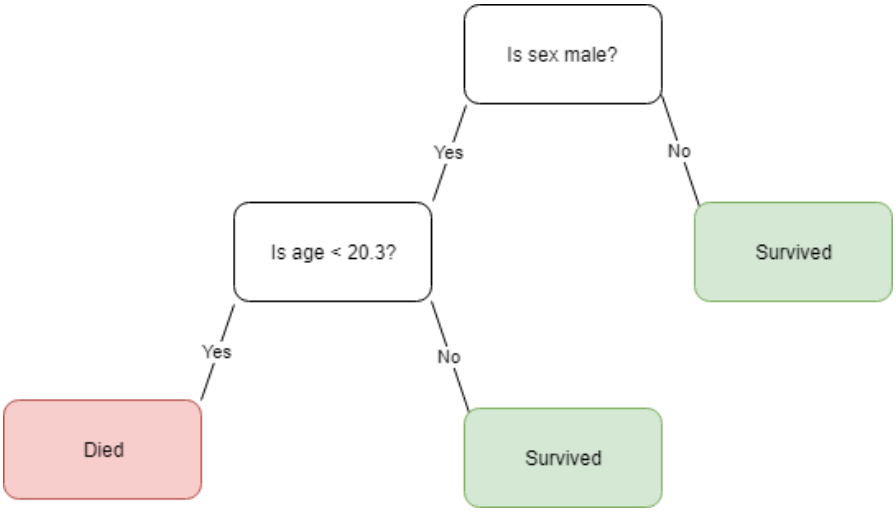

Figure 2.1 – Example of a decision tree which classifies a passenger to survive a car crash or not.

Decision tree algorithms are allowed to capture non-linear patterns in the training data, which makes them flexible, but also prone to overfitting [24]. Kotiantis [22] claims that there are two common solutions to battle overfitting in decision tree induction algorithms:

1. Stop the training before it fits the training data perfectly
2. Prune the decision tree

The second method has shown to be more successful in practice than the first one [25]. Pruning is a process in which subtrees of the decision tree are replaced by leaves. In other words, the size of the tree is reduced. There are different pruning methods that can be used, but exploring such details are not in the scope of this project.

Among several decision tree induction algorithms, CART and C4.5 are the most commonly used [25]. CART can solve both regression and classification problems and it is available in Scikit-learn [26].

2.2.2 Instance based learning

Instance-based learning (IBL) algorithms store provided training data, which is used at a later point to predict/classify an observation. In other words, when a new observation is received, an IBL algorithm retrieves related observations from memory and uses them to predict/classify a new observation. This behavior means that IBL algorithms process training data quickly, while the classification/prediction process takes more time when compared to, for example, decision trees [27].

An example of an IBL algorithm is k-Nearest Neighbor (kNN). Generally, observations can be considered to be points in an n-dimensional space. The algorithm kNN is

based on the principle that observations in a dataset are generally close to other observations with similar properties. For a new observation, the algorithm finds its k nearest neighbors, and uses the properties of the k nearest neighbors to classify the new observation. kNN can be used to solve both regression and multiclass classification problems in Scikit-learn [28], [29].

Wu et al. [30] claim that kNN is suitable for multiclass classification problems. Another study by L. Zhong et al. [31] showed success in predicting CT images from MRI data by using kNN regression. However, [30] mentions that the algorithm is sensitive to the choice of k: smaller values can mean it is sensitive to noise and larger values may include too many points from other classes. Overall, Okamoto and Yugami [32] showed that an optimal choice of k grew linearly with an increase in the amount of training data. This indicates that k should be higher for higher amounts of training data. However, scientific methods for choosing a specific optimal value of k are lacking [22].

Wu et al. [30] talk about the issues of choosing a distance metric for kNN. They argue that among several choices, it is desirable to have a distance metric in which a smaller value between two nodes implies a stronger connection. Scaling is another factor one needs to consider when using kNN. If one feature f_1 varies from, say 1.5 to 1.8, and another f_2 from 10,000 to 1,000,000 then f_2 will have higher impact on the computation of distance.

2.2.3 Bayesian learning

Bayesian algorithms are those that apply the Bayesian inference theorem. This can be used to calculate the probability of a hypothesis H after seeing the data D.

$$p(H|D) = \frac{p(H)p(D|H)}{p(D)} \tag{2.8}$$

The basic notion of these kind of algorithms is that classification can be achieved if assumptions can be made from existing data.

The Naïve bayes algorithm assumes that features in a dataset are independent from one another. This means that each feature contribute independently to classify an observation, regardless of any correlation that might exist among the features, which is why it is called naïve. Naïve bayes is a parametric algorithm due to the fact that it make assumptions on the dataset. Despite its naïve assumption and ease of use, Naïve bayes classifiers have proved successful, even when strong dependencies among features are present [33], [34]. Naïve Bayes can be applied to regression problems through discretization, but it has limited success in comparison to classification problems when the independence assumption does not hold [35]. Naïve bayes can be applied to multiclass classification problems [36].

2.2.4 Regression based learning

Although the names might be confusing, regression based learning is not the same as regression predictive modelling, whose type of problems are referred to as "regression

problems" throughout this project. Regression based learning refers to regression analysis: a number of statistical methods for estimating relationships among variables. This is a well-known tool both in statistics and machine learning.

There are three components in a regression model: scalars β_i, independent variables X_i and dependent variables Y. In regression based learning, X_i represent input features, Y is the target feature and β_i are parameters which are tuned during the training process. The example shown in equation 2.2 is an example of a model built with a regression based learning algorithm called Ordinary Least Squares (OLS). OLS can be used to solve regression problems. A regression based learning algorithm called Logistic regression can be used to solve classification problems.

Both OLS and Logistic regression are parametric since they make assumptions on the dataset. Some of the assumptions listed by [12] that most parametric regression based learning algorithms make:

- Linear behavior: If a target feature and an input feature have a linear relationship, they have straight line when plotted against oneanother. Any non-linear relationships between the target feature and other features are assumed to not be used as input features.
- Normal distribution: Assumes the data of the target feature is normally distributed.
- Outliers: Are expected to be taken care of.
- Data accuracy: Values of the target feature that are irrelevant for classification/prediction are assumed to be deleted. For example, error states that are not to be classified are assumed to be deleted.

Overfitting in regression-based learning algorithms can be thwarted by applying the regularization techniques as seen in section 2.3.2.

2.2.5 Artificial Neural Networks

Artificial Neural Network (ANN) algorithms take inspiration from biological neural networks found in the human brain. An ANN is a directed weighted graph where each node represents an artificial neuron. The graph is organized from left to right by having three type of nodes:

- Input nodes
- Hidden layer nodes
- Output nodes

The input nodes are the different input features and the output nodes are possible outcomes. Without going into too much detail, the purpose of the hidden layers is to combine the inputs and project them on a higher dimension. For example, imagine a

neural network model trained to recognize triangles. The input nodes represent all of the different pixels. The first hidden layer of that model may distinguish edges in the picture, the second layer recognizes when those edges form complex shapes, and so on. Figure 2.2 depicts an example of an ANN with the same kind of example used to explain how CART works (see figure 2.1).

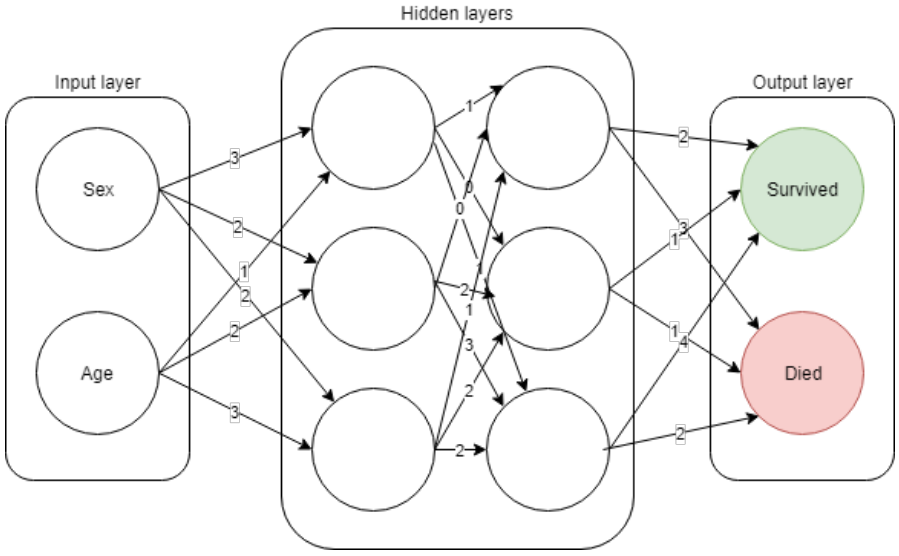

Figure 2.2 – Example of a neural network with two hidden layers. The goal is to classify if a passenger survives a car crash or not.

The number of hidden layers h and nodes per hidden layer N_h are hyperparameters of ANNs. Hyperparameters are model parameters whose value are set before the training process. So the question one might ask is, what is an good choice for h and N_h? Heaton [37] writes that having $h = 1$ is enough for most practical problems, and that in theory, there is little reason to use more than two hidden layers. As for N_h, Heaton writes that a low value of N_h can lead to underfitting whereas the opposite may introduce overfitting and a significant increase in training time. Heaton encourages practitioners to try different values for N_h but provides several rule-of-thumb methods for simplicity, one of which is the following: "The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer".

All that's been covered so far of ANNs is the architecture, but how are decisions made? An ANN algorithm known as Multi-layer Perceptron (MLP) is used as an example to explain how choices are made in the ANN: A specific decision is made from the highest valued output node. The value of a specific non-input node is based on an activation function and edge weights from incoming edges. A technique known as backpropagation is used in MLP to set the edge weights. In a simple sense, backpropagation updates weights of edges in the ANN based on a loss function. Whenever a classification/prediction is made, the error is propagated backwards in the network from that classification/prediction, and weights are adjusted accordingly. MLP is a supervised

learning algorithm that is available in Scikit-learn [38].

2.2.6 Ensemble learning

Ensemble learning is based on the principle that combining the results of several algorithms into a collective score can produce better results than individual scores. One example of an ensemble learning algorithm Random forest. Random forest averages results from several decision tree classifiers/predictors. Two independent studies indicate that Random forest generally have high performance when compared to other non-ensemble algorithms [39], [40]. Random forest can be applied to classification- and regression problems [28]

2.3 Generalization

The fundamental goal of machine learning is to generalize beyond observations in the training dataset, since it's unlikely that the same exact observations are found again on unseen data [41]. Both training error: how well an algorithm performs on its training data, and generalization error: how well a model performs on unseen data, need to be considered in machine learning [11].

The terminology used to explain how well machine learning models learn and generalize to new data is overfitting and underfitting. They are two central challenges in machine learning [11].

- Overfitting: Random fluctuations and statisticical noise is learnt to the extent that it affects the model's ability to generalize. Instead of learning the data trend in the training data, the model "memorizes" it [13]. An overfit model is overly complex.
- Underfitting: A model that performs poorly on both its training data and on generalization. An underfit model is too simple.

The goal then, is to select a model that is somewhere between underfitting and overfitting. Underfitting is typically remedied by choosing alternative models, but the most common problem in applied machine learning is how to avoid overfitting [42]. As a means to check whether or not a model suffers from overfitting, [43] suggests that when the test error of a model exceeds its training error, the model is overfitted to its training data. According to Davide [13] the mere awareness of the issue of overfitting along with two powerful tools: cross-validation and regularization, can be enough to overcome the problem. Feature selection is also brought up in this section as a means to overcome overfitting.

2.3.1 Cross-validation

An alternative to the holdout approach explained in section 2.1 is Cross-validation. It is an approach where parts of the data are not necessarily used solely for testing, it can be

used for both training and testing. One cross-validation technique is called k-fold cross-validation. With this technique, the data is randomly shuffled and split into k folds. The idea is to iterate the training- and test process k times so that every fold has been used once for testing, and ultimately average the performances from k iterations. Using a value of $k = 10$ is a common choice in practice and in which case it is called 10-fold cross-validation [13]. Furthermore, the results from [44] indicate that using $k = 10$ is a good choice when it comes to minimizing running-time, limiting bias (underfitting) and variance (overfitting).

While this technique can be used to limit overfitting, it also proves useful when dealing with small datasets since all of the data can be used for training [13]. On the other hand, a different study shows that since k-fold uses its data both for training and evaluation, it isn't entirely unbiased and that it can create naïve models [45].

There is a variant of this technique called stratified k-fold cross-validation. In stratified k-fold cross-validation, the folds are created in such a way that each fold contains similar proportions of target features as the full dataset. For example, think of the classification problem of classifying email as spam or not spam. If this technique is applied to the email filtering problem as seen in 2.1.1, and the ratio of spam/not spam is 20%/80% in the original dataset, then the same proportion is attempted to be maintained in each of the k folds. This technique tends to generate less bias and variance when compared to regular k-fold cross validation [46]. It can be applied to regression problems, but the results from Breiman and Spector [21] indicate that there is little improvement in doing so.

2.3.2 Regularization

Another method used to overcome overfitting is regularization. This technique discourages complexity and flexibility of models by regularizing its coefficients toward zero. The magnitude of the regularization can be controlled by a hyperparameter λ. The higher value of λ, the higher impact regularization has on the model. High values on λ can result in underfitting and should therefore be controlled carefully [47]. Three different types of regularization methods:

- L_1 regularization (Lasso): Adds a penalty equal to the sum of the absolute values of n coefficients. This kind of regularization can nullify parameters and for that reason it can be seen as a way to limit the amount of features in the model (see feature selection in 2.3.3).

$$Error_{L_1} = Error + \lambda \sum_{i=1}^{n} |\beta_i| \tag{2.9}$$

- L_2 regularization (Ridge): Adds a penalty equal to the sum of the square value of the coefficients. This exhibits a different behavior than L_1 regularization in that

the coefficients are reduced to zero more slowly.

$$Error_{L_2} = Error + \lambda \sum_{i=1}^{n} \beta_i^2 \tag{2.10}$$

- L_1/L_2 regularization (Elastic-net): A combination of L_1 and L_2 regularization. A hyperparameter α is set to determine the impact ratio of L_1 and L_2 regularization, where $0 \leq \alpha \leq 1$.

$$Error_{L_1L_2} = Error + \lambda((1-\alpha) \sum_{i=1}^{n} |\beta_i| + \alpha \sum_{i=1}^{n} \beta_i^2) \tag{2.11}$$

A comparison of these techniques was made by [48] in their performance of modelling insulin sensitivity. The results demonstrate a slight advantage to Lasso and Elastic-net in terms of performance.

2.3.3 Feature selection

Feature selection is a process where irrelevant and redundant features are removed. The curse of dimensionality, which is brought up in section 1.1, can be avoided by reducing the amount of input features. Reducing the amount of input features can lead to better generalization and performance [49], [50]. There are methods in Scikit-learn to perform feature selection based on linear correlation to a model's target feature, removing features with low variance, and more [51].

2.4 Optimizing hyperparameters

There exist no solution for finding optimal hyperparameters for machine learning algorithms a priori. Instead, experimentation is needed. One brute-force approach, which according to [52] is the best way to find optimized hyperparameters, is to use a technique known as grid search. The technique systematically test hyperparameters incrementally in a given interval. Although the technique may prove useful, it can be computationally intense.

Another way to test parameters is to use a technique known as random search. As the name suggests, it tests values at random and picks the best choice from a number of random tests. According to [52], this technique works surprisingly well, but the authors recommend to perform at least 15 to 60 tests in order for it to work properly.

2.5 Data preparation

Data preparation encompass different measures used to update a given dataset. It may involve removing errors, transforming features, scaling features etc. Davide [13] states that the most important component of a machine learning project is the dataset, and that the success of a machine learning project relies on how the dataset is pre-processed.

2.5.1 Imbalanced data

Imbalanced data is covered briefly in section 2.1.1, in short it means that the classes of a specific feature in a dataset are not equally represented. This may be a concern in classification problems. Fortunately, there are a number of ways to overcome this problem. Davide [13] suggests that the best way to overcome it is to collect more data. Apart from collecting more data, there is also a method known as oversampling that can be used. The method involves creating new observations from existing ones with underrepresented classes.

A number of oversampling techniques that can be used to solve multiclass classification problems are suggested by [53]. However the suggested techniques are complex and must be integrated manually to comply with Scikit-learn. Imbalanced-learn is a Scikit-learn compatible Python module that can be used to perform oversampling [54]. Three techniques are supported for oversampling: Smote, Adasyn and Random oversampler. The techniques can be used to solve multiclass classification problems in Imbalanced-learn, despite not being suggested by [53] as proper techniques to solve such problems.

Smote and Adasyn create new synthetic points in the dataset, while Random oversampler creates duplicates. Synthetic points are points that are created from existing points, but with minor adjustments. The disadvantages of using duplicate observations instead of synthetic points is that it introduces overfitting [55]. However, both Adasyn and Smote can be configured in a number of ways on how the synthetic points are built, and are therefore not as easy to use as Random oversampler [56].

2.5.2 Feature engineering

Feature engineering concerns manipulation of features in a dataset, in order to attain an overall improved result. Domingos [41] argues that "feature engineering is the key" in a machine learning project. However, this process can be considered beging more art than science, and as of such, requires human creativity to come up with smart ways to use the given features [52].

Apart from choosing proper features, it might also prove useful to scale down a feature, which operates on a different numeric scale than other features in the same dataset. For example, if a dataset feature shows the number of leaves on a tree, this could be up to 30000 during summer-time and zero during the winter. This could be transformed into a discrete and smaller numeric scale having fewer possibilities using the following transformation:

$$[0, 9999] = \text{few} = 0$$
$$[10000, 19999] = \text{medium} = 1$$
$$[20000, 30000] = \text{many} = 2$$

While a simplified representation in this way can be helpful, it can also mean the representation is too simplified. It could be that more classes are needed to represent leaves in the example presented above. For example that three classes isn't enough to

capture the characteristics of leaves on a tree, but more classes are needed:

$$[0, 5999] = \text{very few} = 0$$
$$[6000, 11999] = \text{few} = 1$$
$$[12000, 17999] = \text{medium} = 3$$
$$[18000, 23999] = \text{many} = 4$$
$$[24000, 30000] = \text{very many} = 5$$

Chapter 3

Method

The chapter covers strategies and methods used to achieve the objective of this project. Reasons for each choice of method or strategy are motivated in the chapter.

3.1 Choice of machine learning software

The author decided at an early stage to work with Scikit-learn, mainly due to the fact that the author has previous experience working with Python, and that Scikit-learn is claimed to be easy to use [57]. Scikit-learn have been used in peer-reviewed machine learning work, which is an indication of it being a legitimate software to use for such purposes [58], [59].

3.2 Literature study

The following databases were used to find relevant books and academic papers to fulfill the aim of the study: Google Scholar, Google Books, Google, Scopus and IEEE Xplore. At times, especially in the beginning of the project, it proved difficult for the author to learn machine learning concepts solely from academic resources. Therefore, non-peer-reviewed resources such as blogs and websites etc. were visited occasionally. If the author learned new concepts from non-peer-reviewed sources, their validity was checked in peer-reviewed material. A number of key-words were used to find relevant material. Some of the key-words used:

- machine learning
- supervised learning
- multiclass classification
- regression
- cross-validation

- methodology
- overfitting
- performance
- model
- sensors
- road condition
- road weather information system

The auhtor attempted to find related work in modelling RWIS sensors but was unable to do so.

3.3 Data overview

The data provided by Trafikverket as seen in section 1.4 was studied from an early stage in the project. This was done primarily to determine if the subtask target features should be modelled by regression or classification algorithms, but also to study the data distribution of each target feature to reveal how often errors occur, if the dataset is imbalanced etc.

Although what some may regard as a big dataset was provided, the author was not sure if it was big enough to fulfill the aim of the study. Furthermore, the provided dataset is geographically limited to a region in western Sweden, and the author doubted that any algorithm trained on its data could perform well in for example, predicting/classifying observations from the northern parts in Sweden. Even so, the RWIS stations which provided the data for this project appear to be the only ones in possession of Trafikverket that have both the DSC111 and DST111 sensors. As of such, it was decided to not investigate if data from additional RWIS stations should be added. When it comes to the size of the dataset, it was decided to see if the project subtasks can be solved using the given dataset, and to find additional data from the given weather stations if needed.

3.4 Research approach

Among all of the available supervised learning algorithms able to solve the project subtasks, finding optimal ones may prove cumbersome. One approach could be to study the benefits and advantages of each algorithm or algorithm family in theory, and assume which one is best for solving each of the project subtasks. In research approach theory, this is known as a deductive approach; where one makes theoretical hypothesis and assumptions, and test and/or verify these empirically. Microsoft Azure and Scikit Learn provide intuitive guides on which algorithm to choose depending on the dataset size and type of problem to be solved [60], [61]. However, similar algorithm-choosing guidelines

seems to be few in academic resources. The author assumes that this is because algorithm performance is affected by a number of variables such as dataset size, amount of input features, choice of hyperparameters and more.

Instead of relying heavily on theory on which algorithm is best suited for a given situation, the author decided that a number of pre-determined algorithms should be tested on different situations. The author assumes that algorithms from the same algorithm family yield similar results. Thus, if at least one algorithm from each algorithm family is represented, well performing algorithms can be found that best solves each of the project subtasks. The process of comparing a number of algorithms, all of which are running on default settings to get a quick assessment, is termed "spot-checking" by [62]. It is deemed necessary in some situations by [12] to run multiple models on the same training and test dataset in parallel and compare their performances to choose an appropriate algorithm for a given problem. Including one additional algorithm for comparison in Scikit-learn means one extra line of code. This makes spot-checking viable with Scikit-learn.

3.4.1 Choice of algorithms to evaluate

The algorithms to evaluate in this project were chosen based on their interpretability, availability in Scikit-learn, ability to solve regression and classification problems, their general popularity, need to represent a certain algorithm family and need to balance the amount of parametric/non-parametric algorithms in total.

Table 3.1 – The supervised algorithms that are evaluated in this project.

Name	Family	Solves	Parametric or non-parametric
CART	Decision tree based learning	regression/classification problems	non-parametric
kNN	Instance based learning	regression/classification problems	non-parametric
Naïve Bayes	Bayesian learning	classification problems	parametric
OLS	Regression based learning	regression problems	parametric
Logistic regression	Regression based learning	classification problems	parametric
Lasso	Regression based learning	regression problems	parametric
MLP	Deep learning	regression/classification problems	non-parametric
Random forest	Ensemble learning	regression/classification problems	non-parametric

CART was chosen since it is the only decision tree based learning algorithm available in Scikit-learn. The author chose kNN, Naïve Bayes, MLP and Random forest based on their abilities to solve regression and classification problems, and based on a hypothesis made by the author that they are among the most popular algorithms within their families. Two algorithms were chosen to represent regression based learning algorithms: OLS and Logistic regression. OLS and Logistic regression were both chosen since they are parametric, generally popular and that either of them cannot solve both regression and classification problems.

Regularization techniques, which are covered in section 2.3.2, are used to penalize an algorithm to minimize overfitting, and in some cases perform feature selection. However, in Scikit-learn they can be used like a stand-alone algorithm: For example, the built-in Lasso regularization in Scikit-learn is essentially a penalized version of OLS [63]. It was

decide to use at least one regularization technique as a stand-alone algorithm as part of the spot-checking. Lasso was chosen instead of Ridge or Elastic-net as regularization techniques since theory from section 2.3.2 suggests a slight advantage to Lasso and Elastic-net, and Lasso was ultimately chosen because of its ability to perform feature selection and the fact that it has one hyperparameter instead of two. This makes it easier to optimize hyperparameters with Lasso (see Lasso optimization in 3.5.1).

Table 3.2 and 3.3 show the algorithms used to solve regression- and classification problems in this project. The algorithms are used with their default Scikit-learn settings in spot-checking experiments.

Table 3.2 – The supervised algorithms, which solves regression problems, that are evaluated in this project. Default settings for relevant hyperparameters are shown.

Name	Hyperparameter 1 default setting	Hyperparameter 2 default setting
CART	splitting critera: gini	
kNN	$k = 5$	distance metric: minkowski
OLS		
Lasso	$\lambda = 1$	
MLP	n.o. hidden layers: 1	n.o. hidden nodes: 100
Random forest	n.o. estimators: 10	

Table 3.3 – The supervised algorithms, which solves classification problems, that are evaluated in this project. Default settings for relevant hyperparameters are shown.

Name	Hyperparameter 1 default setting	Hyperparameter 2 default setting
CART	splitting critera: gini	
kNN	$k = 5$	distance metric: minkowski
Naïve Bayes		
Logistic regression	penalty: L_2	
MLP	n.o. hidden layers: 1	n.o. hidden nodes: 100
Random forest	n.o. estimators: 10	

3.5 Research strategy

An experimental research strategy was used to solve the subtasks of this project. The general goal of an experimental research strategy is to study the cause and effect relationship among variables in an experiment. The variable under consideration is called the independent variable, and the general idea is to study an overall effect of varying the independent variable.

3.5.1 Experimental setups

To avoid repetitive explanation of experimental setups, and to keep some parameters constant throughout experiments, a number of experimental setups were set and named by the author of this project. They are as follows:

- Holdout regression spot-checking: This experiment tests all available regression algorithms in table 3.2, where each algorithm use its default settings. Holdout is used as a validation technique with which the dataset is split into 80% training and 20% testing. A random state is set to shuffle the dataset before splitting, but to maintain deterministic behavior. The random state is obtained by using a seed, where $seed = 7$ is an arbitrary choice.

- Holdout classification spot-checking: Same as the setup mentioned above but using the classification algorithms in table 3.3.

- Cross-validation regression spot-checking: Uses a similar setup as holdout regression spot-checking but with k-fold as a model validation technique instead of holdout. A value of $k = 10$ is used since it is supported in theory from section 2.3.1.

- Cross-validation classification spot-checking: Similar to forementioned setup, apart from using stratified k-fold instead of regular k-fold. The classification algorithms in table 3.3 are used.

- kNN optimization: The purpose of this experiment is to find optimized choice of k in kNN. As brought up in section 3.7, grid search is used as hyperparameter optimization technique, which applies to this experiment as well. Theory from section 2.2.2 suggests that there is no formula for providing an optimal value for k, and therefore an interval of values are tested, including the default value of $k = 5$. The interval of values for k is chosen based on generally popular choices found by the author during the literature study.

$$k = 2^i \qquad 0 \leq i \leq 6 \tag{3.1}$$

  A maximum of $i = 6$ is set to minimize the running time of this experiment.

- MLP optimization: This experiment deals with finding a good choice for number of hidden nodes in MLP. Similar to the choice of k in kNN, a good choice of number of hidden nodes in MLP seems to be somewhat of a mystery. As mentioned in section 2.2.5, a starting point is to set the number of hidden nodes n_h to $n_h = \frac{2}{3}(n_i + n_o)$ where n_i and n_o are number of input and output nodes respectively, which corresponds to input- and target feature(s). The number of input features vary in this project, and as of such, the forementioned rule of thumb is used as a guidline to create a numeric interval to test different values for n_h.

$$n_h = 4^i \qquad 0 \leq i \leq 4 \tag{3.2}$$

The default value in Scikit-learn is $n_h = 100$, which is also included in the grid search.

- Lasso optimization: The purpose of this experiment is to find an optimized value for λ in Lasso. Apart from the standard value $\lambda = 1$, a range of values are tested. The interval of values for λ is chosen based on generally popular choices found by the author during the literature study.

$$\lambda = \frac{100}{10i} \qquad 0 < i < 6 \tag{3.3}$$

3.6 Choice of model validation technique

The way overfitting is detected in this project is to compare how a fitted model performs on the training dataset to its performance on the test dataset. The author did not find a way to achieve this using the built-in k-fold or stratified k-fold in Scikit-learn, which is one of the main reasons why holdout was preferred as model validation technique over cross-validation. Another reason why holdout was used instead of k-fold, or stratified k-fold, was that it has shorter running-time. This is most likely because k-fold essentially performs k model fits, whose performances are averaged when finished, while holdout does one model fit. Although cross-validation was not used in achieving the final results, it was used in the early stages of the project, and during the data preparation process, due to the fact that it was originally planned to be used to attain the project results.

When k-fold was used, a value of $k = 10$ for both k-fold and stratified k-fold was used since this is supported in theory on cross-validation from section 2.3.1. Whenever cross-validation techniques were used, stratified k-fold was used for classifying precipitation type to ensure that each class was equally represented in the folds, and regular k-fold was used for regression problems. When holdout was used as model validation technique, a 80%/%20 training/test split was used. This is an arbitrary choice, taken from the assumption that an optimal split does not seem to exist in theory (see theory on holdout in 2.1).

3.7 Choice of hyperparameter optimization technique

Randomized search was not used in this project because it elicits randomness. Since an experimental approach was used in this project, it was deemed necessary to maintain control over dependent variables, and to ensure that several runs on the same settings produce the same results, i.e. the desired behavior of the spot-checking is to be deterministic. It was decided to use grid search, which may be more demanding computation-wise than Randomized search. This was mitigated by limiting the interval of values to test, and to limit the amount of hyperparameters to optimize. Three hyperparameters in total were optimized: k in kNN, n.o. hidden neurons in MLP and λ in Lasso (see project delimitations in 1.3).

3.8 Choice of performance measures

The author chose to use MSE to evaluate the performance of regression algorithms since theory on regression performance measures suggests that it is essential to use MSE for regression problems (see theory on regression in 2.1.2). Performance of classification algorithms were measured using $\overline{F1}$ score. It was chosen due to the fact that it takes both precision and recall into account, and that theory from section 2.1.1 suggests that using accuracy alone can be misleading for imbalanced datasets, which is the case of precipitation type. Accuracy was not used to evaluate classification algorithms, but displayed nevertheless, since it can give an intuititive understanding of how an algorithm performs.

As described in section 1.2, the project subtasks are about finding algorithms that performs best in terms of both performance and generalization. How performance is measured has already been covered in this section, whereas generalization has not. As theory suggests in section 2.3, overfitting is when a model fits its training data significantly better than its test data. Theory from section 2.3 indicate that underfitting is the result of a poor model fit. The author hypothesize that underfit models are ruled out as poor performers in this project, which is why focus was set on overcoming overfitting. It was decided to quantify overfitting rather than making it binary: overfitting/no overfitting. It was quantified by calculating the difference of how a model performs on its test dataset P_{test} in contrast to its training dataset P_{train}. Higher values in equation 3.4 indicate overfitting.

$$\text{Overfitting} = P_{test} - P_{train} \tag{3.4}$$

3.9 Experimental methodology

This section describes the methodology used to obtain the results of this project. The first step involves preparing the data, which is followed up by obtaining the results to solve the project subtasks.

3.9.1 Data preparation

Pyle [64] argues that data preparation "is not a process which can be carried out blindly". He claims that an arbitrary dataset cannot be fixed by an automated data preparation process. This indicates that there is not one correct way to carry out a data preparation process. The author selected methods from three academic resources [64]–[66] that are thought to be relevant steps in the data preparation process for this project. The process was done in the following way:

1. Data cleaning: It was revealed during the early stages of this project that some of the sensors elicited a considerable amount of errors. As mentioned in the project delimitations (see delimitations in 1.3), this project is about modelling non-error behavior, which is why any reported errors are removed at this step. There may

also be suspected errors such as outliers that are considered for removal at this step.

2. Feature selection: The idea of this step is to see if any input features should be ruled out due to irrelevance or negative effect on overall performance. Suspected features which are believed to have a negative effect on overall performance are tested to see how they affect overall performance.

3. Data transformation: This step involves dealing with potential class imbalances and to see if feature engineering can be employed to improve overall performance. Oversampling techniques (see theory on imbalanced datasets in 2.5.1) are tested to see if they can be used to mitigate the effect of imbalanced datasets.

3.9.2 Obtaining the results

After the data preparation process, the strategy used to find the best performance in predicting/classifying target feature f_t in each subtask in this project is as follows:

1. Rank the possible input feature f_i to f_t. This is done by calculating a correlation score with f_i to f_t. The correlation score is calculated with a so-called F-value, which estimates the degree of linear dependency between any two variables [67], [68]. The purpose of this step is to avoid testing all possible combinations of input features, in finding a good choice of input features to use in the spot-checking step.

2. Run n spot-checking experiments on the algorithms that can classify/predict f_t, where n is the amount of input features available in total. The idea is to find which input features give the best performance- and generalization score for each algorithm. The ith spot-checking run uses the top i input features, where $1 \leq i \leq n$.

   For regression tasks, holdout regression spot-checking experiments are used (see experimental setupts in 3.5.1). Performance is measured in MSE on the test dataset MSE_{test}. Overfitting MSE_{diff} is measured by calculating the difference of how a model performs on the test dataset, in contrast to how it performs on the training dataset: $MSE_{diff} = MSE_{test} - MSE_{train}$. To attain an overall score which covers both performance and overfitting, an accumulated score $P_{acc} = MSE_{test} + MSE_{diff}$ is calculated. A minimal score of P_{acc} is desired for regression tasks.

   For classification problems, holdout classification spot-checking experiments are used (see experimental setups in 3.5.1). Performance is measured in $\overline{F1}$ score on the test dataset $\overline{F1}_{test}$. Overfitting $\overline{F1}_{diff}$ is measured by calculating the difference of how the model performs on the test dataset in contrast to how it performs on the training dataset: $\overline{F1}_{diff} = \overline{F1}_{test} - \overline{F1}_{train}$. An overall score, which covers both performance and overfitting, is displayed as an accumulated score $P_{acc} = \overline{F1}_{test} - \overline{F1}_{diff}$. While a minimum P_{acc} score is desired for regression tasks, the opposite is true for classification: a higher P_{acc} scores means a better score. Accuracy is also displayed, but not evaluated on.

3. Once a desired set of input features are established for algorithm a_k to predict/classify f_t, specific hyperparameter settings of a_k are varied to see if its performance can be improved further. As mentioned in the project delimitations (see delimitations in 1.3), only k in kNN, number of hidden nodes in MLP and λ in Lasso are optimized. Lasso, MLP and kNN optimization experiments are run as described in section 3.5.1.

4. Lastly, the optimized scores of each algorithm are listed. The algorithm, whose accumulated performance score is best, is identified as the top performing algorithm in predicting/classifying f_t.

3.10 Tools

A number of tools were used to achieve the results in this project:

- Scikit-learn: Python machine learning library. Why Scikit-learn was used is motivated in section 3.1.
- Python: Programming language.
- Pandas: A Python library used in this project to read the dataset from Microsoft Excel workbooks.
- Pyplot: A Python library used to visualize datasets.
- Imbalanced-learn: A python library with functionality to deal with imbalanced datasets.
- Microsoft Excel: Microsoft software. The provided dataset was stored in separate Excel workbooks which is why built-in Excel tools were used to visualize the dataset and calculate mean values, number of occurrences etc.
- Visual Basic: A programming language that can be used in Microsoft Excel to create custom functionality and so-called macros. This was primarily used in the data cleaning and data transformation steps (see 4.1 and 4.3) which could potentially take long time to perform manually.
- Sublime text: Source code editor. Used since the author had previous experience working with Sublime.
- Git: Version control tool. Used to backup the project files, as well as to have the ability of going back to previous versions in the code and other project files.

Chapter 4

Data preparation process

This chapter describes the process of preparing the dataset with data cleaning, feature selection and ultimately data transformation.

4.1 Data cleaning

In the dataset documentation provided by Trafikverket (see provided data in 1.4), there is a note on the measurements from station 1429 saying: "Unreasonable DST111 measurements from about 2016-11-15 to 2016-12-31". It was found in the measurements from station 1429 that the average difference between the DST111 and TIRS road surface temperature measurements were 1.71°C whereas in stations 1402 and 1431, which are the two geographically closest stations to 1429, the average differences were 0.614°C and 0.54°C. In some cases, the difference between the measurements from the two temperature sensors in station 1429 were more than 40°C. This indicates that something may have been wrong with DST111, or some other instrument, at the time. Therefore, observations from 2016-11-15 to 2016-12-31 were removed from the workbook containing data from station 1429. This resulted in an average road surface temperature difference of 0.92°C, which is still higher than the two nearby stations, but whether this was unreasonable or not could not be determined by the author.

In addition to removing suspicious outliers in the dataset, there are also cases where errors are explicitly reported by the sensors. As mentioned in the project delimitations (see delimitations in 1.3), only non-error behavior is modelled in this project. As of such, every observation where at least one error is reported by any sensor, were removed. Figure 4.1 shows that the DSC111 alone was malfunctioning for unknown reasons in more than 50000 observations.

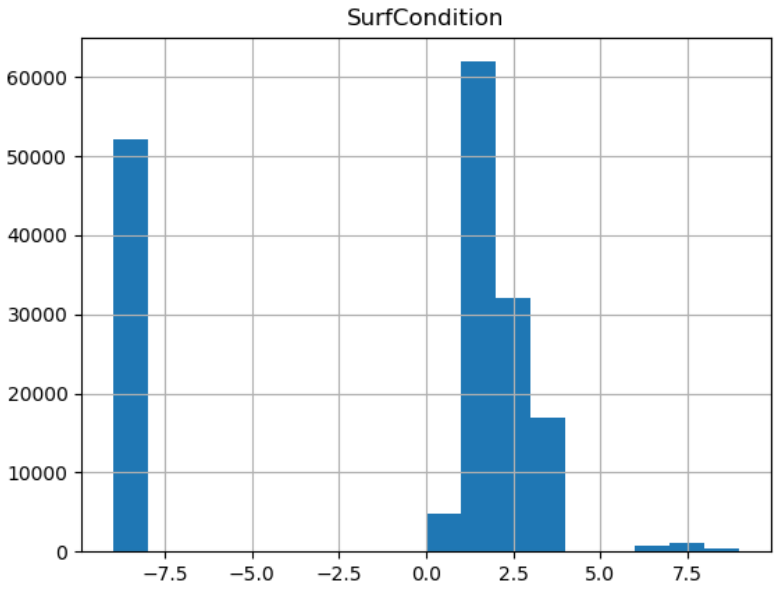

Figure 4.1 – Histogram showing the distribution of the different surface condition types (see table 1.2 to see what each code means).

More suspicious outliers may be present in the dataset that could potentially be identified by applying, for example, a confidence interval. But it was decided to not investigate this matter further since it was assumed that analytical knowledge in road condition data is needed to decide if an outlier represents an error or a correct abnormal value. After the data cleaning process, a total of 115180 observations remained, which means 56245 observations were removed.

4.2 Feature selection

It was decided to investigate if timestamp should be excluded as a possible input feature, primarily because its values are higher than the rest of the features. Theory from section 2.2 suggests that some supervised learning algorithms, such as kNN, are sensitive to scaling. Timestamp is represented in the following format: mmddhhmm, which is interpreted as integers by any model that use it as an input feature. For example, an observation from station 1520 from 12/31/2016 23:30 have the following values: timestamp = 12312330, TIRS surface temperature = 7.1, . . . friction = 0.74.

Although the scaling of the timestamp feature is significantly different from the other input features, it seems to be linearly correlated with other features. Figure 4.2 shows how every feature is related to one another, it indicates that features such as road surface temperature and friction are linearly correlated with timestamp. The correlation may come as no surprise since the northern hemisphere is colder during winter-time, which

means colder road surface temperatures and lower friction, and the other way around during warmer periods.

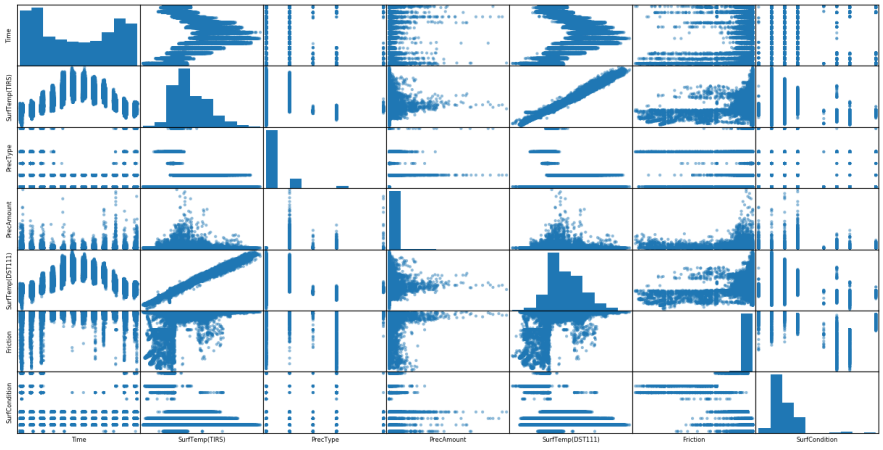

Figure 4.2 – Depicts how the different features used in this project are correlated to oneanother.

To test the relevance of using timestamp as input feature, an experiment was carried out to predict TIRS road surface temperature. Once using timestamp as input feature, and once without. The experiment runs the Cross-validation regression spot-checking setup (see experimental setups in 3.5.1). Table 4.1 shows the result from the experiment.

Table 4.1 – Experiment to see if timestamp is relevant to use as input feature.

Algorithm	MSE not using time	MSE using time
OLS	1.23	1.22
CART	1.21	1.84
kNN	1.25	4.69
MLP	1.06	55663.45
Lasso	1.25	1.24
Random forest	1.13	1.26
Average total	1.19	9279.13

The results indicate that OLS and Lasso achieve slightly better performance by using time as input feature, whereas CART, kNN, Random forest and especially MLP, suffer in terms of performance when doing so. This is an indication that overall performance is improved by not using time as input feature, but the possibility of using timestamp as input feature was not ruled out yet. Section 4.3 deals with testing if timestamp can be scaled down to similar levels of other features and see if it improves overall performance

or not.

4.3 Data transformation

4.3.1 Transforming timestamp

Table 4.1 shows significant improvement in overall performance in not using timestamp as an input feature. However, figure 4.2 shows that timestamp may be a relevant feature to include in that it shows linear correlation with other features. It was decided to test if a transformation of the timestamp feature could yield better performance than using it in its original form. The transformation involves using only month, and time of day from the timestamp feature, but to separate it into two columns so that they operate on lower numeric intervals. Figure 4.3 shows the transformation.

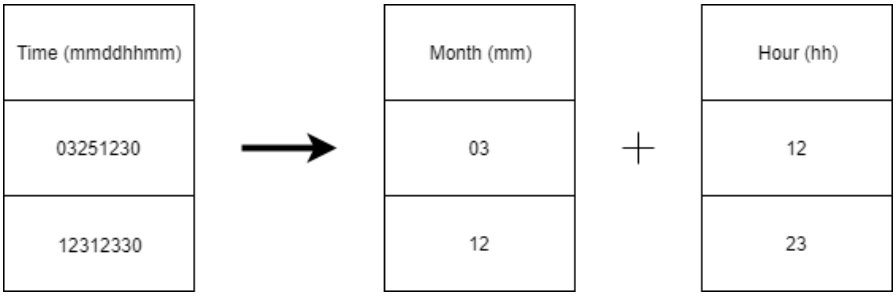

Figure 4.3 – Shows how timestamp as feature is transformed to two new features: month and hour.

The author assumes that month and time of day affect changes in weather conditions more than separate days within a month. Figure 4.4 show the correlations among the features with month and hour used instead of timestamp.

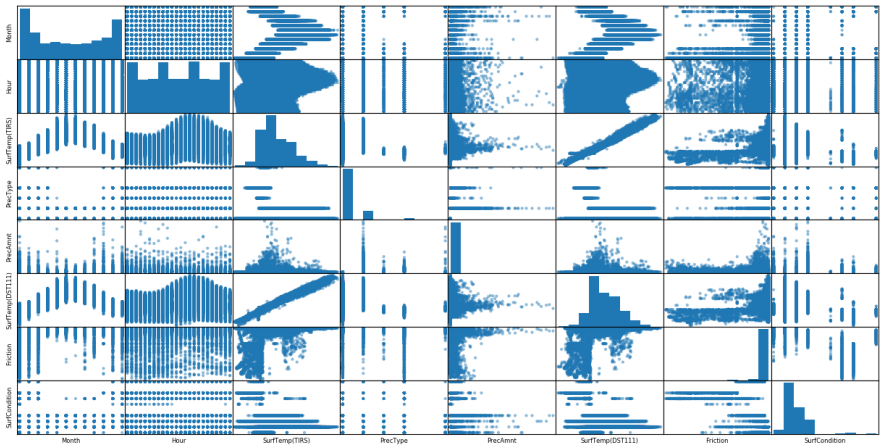

Figure 4.4 – Correlations among the different features where month and hour are two new features that have replaced timestamp.

An experiment to predict TIRS road surface temperature with Cross-validation regression spot-checking setup was used to study the effects of the new transformation.

Table 4.2 – Experiment to see the effects of transforming the timestamp feature to two new features: month and hour.

Algorithm	MSE using neither timestamp nor new features	MSE using timestamp	MSE using hour and month	MSE using month	MSE using hour
OLS	1.23	1.22	1.22	1.22	1.21
CART	1.21	1.84	1.77	1.34	1.39
kNN	1.25	4.69	1.08	1.24	1.15
MLP	1.06	55663.45	1.30	1.06	1.08
Lasso	1.25	1.24	1.23	1.24	1.23
Random forest	1.13	1.26	1.01	1.02	1.12
Total average	1.19	9279.13	1.27	1.19	1.20

The results from table 4.2 indicate that performance is improved for all algorithms when the new features are used as a way to represent time rather than using timestamp. However, not using timestamp, hour or month, seem to have slightly higher performance than using both hour and month, but similar to using either of them. The slight differences in performance in not using any time-related features, as opposed to using the feature engineered ones, could be a a consequence of operating on different dimensionalities. For example MLP shows improved performance when fewer input features are used in table 4.2.

Although little to no overall performance improvement was made using the transformed time features, as opposed to not using any time-related features, it was decided to use the transformed input features as part of default input features for spot-checking experiments. This experiment is set up to test TIRS road temperature alone. The new features may prove more relevant with other target features and specific hyperparameter settings etc.

4.3.2 Handling class imbalance

Figure 4.1 shows that road surface status suffers from class imbalance. But classifying road surface status is not a goal in this project. It is interesting to see if precipitation type have class imbalance as well, since classifying precipitation type is a goal in this project. Figure 4.5 shows the distribution of precipitation type in the dataset.

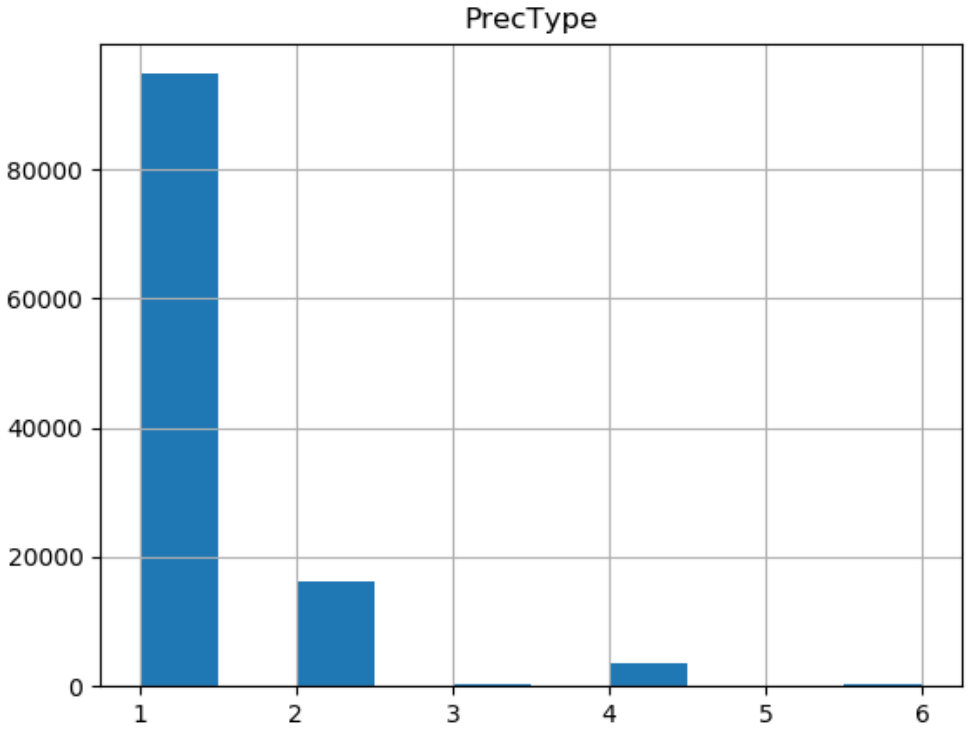

Figure 4.5 – Histogram showing the distribution of the different types of precipitation in the data (see table 4.3 to see what each code means).

The exact number of occurences and a translation of what each code means is shown in table 4.3.

Table 4.3 – Number of occurrences for different types of precipitation.

Code	Precipitation type	N.o. occurrences
1	no precipitation	94825
2	rain with $>= 0°C$ air temperature	16094
3	rain with $< 0°C$ air temperature	266
4	snow	3677
6	rain and snow mixed	316

From what can be seen in table 4.3, precipitation type also have a significant class imbalance. The biggest difference is between no precipitation and rain and snow mixed, the former appears about 300 times more often in the dataset than that the latter.

An experiment was carried out to see if the class imbalance problem can be mitigated by using random oversampling or Smote as oversampling techniques (see information on oversampling techniques in 2.5.1). Smote runs on the default settings as seen in [56]. Both random oversampling and Smote were compared to a scenario where oversampling is not used. All three scenarios run the holdout classification spot-checking setup, to classify precipitation type. Holdout was used instead of cross-validation in the three scenarios since it proved cumbersome to integrate oversampling techniques with cross-validation.

Table 4.4 – Results from not using any oversampling techniques.

Algorithm	Accuracy	$\overline{Precision}$	$\overline{Recall}$	$\overline{F1}$
Logistic regression	0.81	0.24	0.21	0.21
kNN	0.86	0.51	0.35	0.39
CART	0.86	0.53	0.36	0.40
NB	0.82	0.44	0.26	0.26
MLP	0.86	0.47	0.33	0.38
Random forest	0.86	0.56	0.37	0.41
Total average	0.85	0.46	0.31	0.34

Table 4.5 – Results from using random oversampling as oversampling technique.

Algorithm	Accuracy	$\overline{Precision}$	$\overline{Recall}$	$\overline{F1}$
Logistic regression	0.48	0.33	0.55	0.31
kNN	0.57	0.31	0.52	0.31
CART	0.63	0.33	0.54	0.34
NB	0.52	0.36	0.54	0.31
MLP	0.62	0.36	0.60	0.36
Random forest	0.63	0.33	0.54	0.34
Total average	0.58	0.34	0.55	0.33

Table 4.6 – Results from using Smote as oversampling technique.

Algorithm	Accuracy	$\overline{Precision}$	$\overline{Recall}$	$\overline{F1}$
LR	0.48	0.33	0.56	0.31
kNN	0.68	0.31	0.47	0.34
CART	0.72	0.33	0.48	0.35
NB	0.53	0.36	0.53	0.31
MLP	0.62	0.37	0.60	0.36
Random forest	0.72	0.33	0.49	0.36
Total average	0.63	0.34	0.52	0.33

Table 4.4 and 4.5 shows the effects of using random oversampling versus no oversampling: total average accuracy and precision is reduced, total average recall is higher, and total average $\overline{F1}$ score is similar to the case when oversampling is not used. In the case of using Smote as oversampling technique as shown in table 4.6, it proved to have similar effects of using random oversampling.

Both precision and recall are important in this project when it comes to evaulating classification algorithm performance. Since no improvement was made on the collective score of precision and recall: $\overline{F1}$, and overall accuracy was reduced when either of the oversampling techniques were tested, oversampling was ruled out as a possible solution to handle imbalanced classes.

Since the multiclass classification problem is but one of four subtasks in this project, additional effort was not put in to investigate if the imbalanced dataset problem can be solved in other ways.

Chapter 5

Results and analysis

The goal of this chapter is to find answers to the project subtasks. Top performing algorithms for predicting TIRS road surface temperature, precipitation amount and DST111 road surface temperature are identified, as well as the top performing algorithm for classifying precipitation type.

5.1 Experimental setup and summary of results

A solution to each of the project subtasks are covered in the following sections. The results are achieved methodically for each subtask (see experimental methodology 3.9.2). A summary of the methodology of finding an optimized algorithm for predicting/classifying target feature f_t:

1. Rank the possible input features f_i in terms of linear correlation to f_t to get a ranking of the input features. The ranking is referred to as top input features.

2. Run n spot-checking experiments on the algorithms that can classify/predict f_t, where n is the amount of input features available in total. The idea is to find which input features give the best performance- and generalization score for each algorithm. The ith spot-checking run uses the top i input features, where $1 \leq i \leq n$. An accumulated score P_{acc} is calculated which takes both performance and overfitting into account.

3. Optimize the hyperparameters of Lasso, MLP and kNN using their optimized choice of top input features.

4. Analyze the best scores for each algorithm using their optimized choice of top input features and hyperparameter settings. The algorithm whose accumulated performance score is best, is identified as the top performing algorithm.

Table 5.1 shows a summary of the top performing algorithms that were found for modelling each target feature related to the project subtasks.

Table 5.1 – Shows the top performing algorithms for each of the project subtasks.

Sensor	Subtask	Problem type	Best algorithm	Choice of top performing input features	Optimized settings	Best performance
Optic eye	model precipitation type	classification problem	CART	road surface condition road friction DST111 road surface temperature hour month	Scikit default	$(Accuracy, \overline{F1}_{test}) = (0.84, 0.46)$
Optic eye	model precipitation amount	regression problem	kNN	road friction	$K = 64$	$MSE_{test} = 0.31$
TIRS	model TIRS road surface temperature	regression problem	MLP	DST111 road surface temperature road surface condition road friction month hour precipitation type	n.o. hidden nodes: 64	$MSE_{test} = 0.88$
DST111	model DST111 road surface temperature	regression problem	Random forest	road surface condition road friction month precipitation type hour	Scikit default	$MSE_{test} = 10.16$

The sections that follow describe how the results in table 5.1 were obtained.

5.2 Predicting road surface temperature (TIRS)

5.2.1 Input features correlation ranking

Table 5.2 – Relevancy of each possible input feature to the target feature: TIRS road surface temperature.

Relevancy ranking	Input feature	Correlation score
1	DST111 road surface temperature	7688772.49
2	road surface condition	11048.87
3	road friction	6840.20
4	month	4300.00
5	hour	1968.02
6	precipitation type	1784.49
7	precipitation amount	238.91

The correlation ranking in table 5.2 shows that DST111 is the most relevant feature while Optic Eye features are less relevant in predicting TIRS road surface temperature.

5.2.2 Spot-checking

Table 5.3 – Results from spot-checking experiment on the top features for predicting TIRS road surface temperature. The results are shown as a tuple: (MSE_{test}, MSE_{diff}) where MSE_{test} represents performance and $MSE_{diff} = MSE_{test} - MSE_{train}$ shows the degree of overfitting, larger values of MSE_{diff} indicate overfitting.

Algorithm	MSE top 7	MSE top 6	MSE top 5	MSE top 4	MSE top 3	MSE top 2	MSE top 1
OLS	(1.19, 0.02)	(1.19, 0.02)	(1.19, 0.02)	(1.20, 0.02)	(1.22, 0.02)	(1.22, 0.02)	(1.22, 0.02)
CART	(1.36, 1.10)	(1.35, 1.08)	(1.31, 1.03)	(1.05, 0.30)	(1.03, 0.15)	(1.02, 0.09)	(1.01, 0.05)
kNN	(0.94, 0.32)	(0.93, 0.32)	(0.92, 0.31)	(1.08, 0.18)	(1.16, 0.09)	(1.16, 0.06)	(1.21, 0.04)
MLP	(0.90, 0.01)	(0.86, 0.02)	(0.86, 0.01)	(0.97, 0.03)	(1.00, 0.02)	(1.02, 0.02)	(1.01, 0.01)
Lasso	(1.24, 0.02)	(1.24, 0.02)	(1.24, 0.02)	(1.24, 0.02)	(1.24, 0.02)	(1.24, 0.02)	(1.24, 0.02)
Random forest	(1.01, 0.65)	(1.00, 0.66)	(1.01, 0.64)	(1.01, 0.23)	(1.01, 0.12)	(1.01, 0.08)	(1.01, 0.05)

Table 5.4 – Shows an accumulated performance score $P_{acc} = MSE_{test} + MSE_{diff}$. Top results for each algorithm are highlighted. In case of ties, the one using the fewest number of input features is considered optimized.

Algorithm	P_{acc} top 7 features	P_{acc} top 6 features	P_{acc} top 5 features	P_{acc} top 4 features	P_{acc} top 3 features	P_{acc} top 2 features	P_{acc} top 1 features
OLS	1.21	1.21	**1.21**	1.22	1.24	1.24	1,24
CART	2.46	2.43	2.34	1.35	1.18	1.11	**1.06**
kNN	1.26	1.25	1.25	1.26	1.25	**1.22**	1.25
MLP	0.91	0.88	**0.87**	1.00	1.02	1.04	1.02
Lasso	1.26	1.26	1.26	1.26	1.26	1.26	**1.26**
Random forest	1.66	1.66	1.65	1.24	1.13	1.09	**1.06**

The results from table 5.4 show that OLS and MLP have best accumulated scores when using the top five features, whereas kNN is optimized when the top two features are used. CART, Lasso and Random forest have optimized scores in using the top one feature.

5.2.3 Optimizing hyperparameters

Table 5.5 – Shows the effect of optimizing the hyperparameters of kNN, MLP and Lasso using their top input features

Algorithm	Default hyperparameter setting	Optimized setting	Default performance P_{acc}	optimized performance (MSE_{test}, MSE_{diff})	optimized performance P_{acc}
kNN	$k = 5$	$k = 64$	1.22	(1.00, 0.04)	1.04
MLP	n.o. hidden nodes: 100	n.o. hidden nodes: 64	0.87	(0.88, 0.01)	0.89
Lasso	$\lambda = 1$	$\lambda = 0.001$	1.26	(1.22, 0.02)	1.24

As shown in table 5.5, accumulated scores were improved for kNN and Lasso when optimized hyperparameter settings were used. MLP showed a slightly reduced accumulated score using optimized hyperparameters. However, since the default settings were used as well in the optimization process, it is believed that setting number of hidden nodes to 64 is indeed an optimized choice and thus an optimized overall performance was obtained.

5.2.4 Results and analysis

Table 5.6 – Shows the overall optimized settings and performances for each of the algorithms in predicting TIRS road surface temperature. The top performing algorithm is highlighted.

Algorithm	Optimized settings	Best choice of input features	Best performance (MSE_{test}, MSE_{diff})	Best performance P_{acc}
OLS	Scikit default	top 5	(1.19, 0.02)	1.21
CART	Scikit default	top 1	(1.01, 0.05)	1.06
kNN	$k = 64$	top 2	(1.00, 0.04)	1.04
MLP	**n.o. hidden nodes: 64**	**top 5**	**(0.88, 0.01)**	**0.89**
Lasso	$\lambda = 0.001$	top 1	(1.22, 0.02)	1.24
Random forest	Scikit default	top 1	(1.01, 0.05)	1.06

Table 5.6 shows that MLP has the lowest P_{acc} score among all algorithms and is thus the algorithm that performs best in terms of performance- and generalization when it comes to predicting TIRS road surface temperature. MLP have a performance score of $MSE_{test} = 0.88$ which means that in predicting TIRS road surface temperature, MLP was on average off by $\sqrt{0.88} \approx 0.94$°C from the actual temperature measurements in the test dataset. Figure 5.1 shows the relationship between predictions made by MLP on the test dataset and actual values.

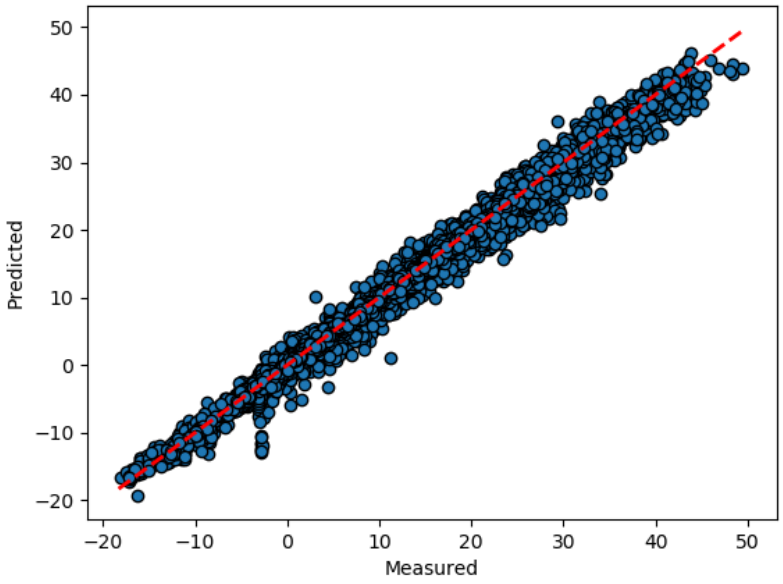

Figure 5.1 – Shows the relationship between actual measurements and predictions made by MLP to predict TIRS road surface temperature. The closer a point is to the red dashed line, the more accurate prediction for that observation.

5.3 Classifying precipitation type (Optic Eye)

5.3.1 Input features correlation ranking

Table 5.7 – Relevancy of each possible input feature to the target feature: precipitation type

Relevancy ranking	Input feature	Correlation score
1	road surface condition	4506.14
2	road friction	4274.88
3	DST111 road surface temperature	1263.89
4	Hour	609.45
5	Month	16.71

The correlation ranking in table 5.7 shows that the road surface condition and road friction are most relevant when it comes to classifying precipitation type whereas the time-related input features are less relevant.

5.3.2 Spot-checking

Table 5.8 – Results from spot-checking experiment on the top features in classifying precipitation type. The results are shown as a triple: (accuracy, $\overline{F1}_{test}$, $\overline{F1}_{diff}$) where $\overline{F1}_{test}$ represents performance and $\overline{F1}_{diff} = \overline{F1}_{test} - \overline{F1}_{train}$ shows the degree of overfitting, larger values of $\overline{F1}_{diff}$ indicate overfitting.

Algorithm	Performance top 5	Performance top 4	Performance top 3	Performance top 2	Performance top 1
Logistic regression	(0.81, 0.21, 0.00)	(0.81, 0.21, 0.00)	(0.81, 0.21, 0.00)	(0.82, 0.21, 0.00)	(0.81, 0.21, 0.00)
kNN	(0.85, 0.40, -0.05)	(0.85, 0.37, -0.05)	(0.86, 0.39, -0.03)	(0.86, 0.34, 0.00)	(0.83, 0.34, 0.00)
CART	(0.84, 0.46, -0.41)	(0.85, 0.44, -0.28)	(0.86, 0.41, -0.06)	(0.86, 0.34, 0.00)	(0.83, 0.34, 0.00)
Naïve bayes	(0.82, 0.27, 0.00)	(0.82, 0.26, 0.00)	(0.82, 0.26, 0.00)	(0.82, 0.25, 0.00)	(0.82, 0.25, 0.00)
MLP	(0.85, 0.35, 0.00)	(0.85, 0.38, 0.00)	(0.86, 0.37, 0.00)	(0.86, 0.34, 0.00)	(0.82, 0.21, 0.00)
Random forest	(0.85, 0.45, -0.40)	(0.85, 0.42, -0.29)	(0.86, 0.42, -0.05)	(0.86, 0.34, 0.00)	(0.83, 0.34, 0.00)

Table 5.9 – Shows an accumalated performance score $P_{acc} = \overline{F1}_{test} - \overline{F1}_{diff}$. Top results for each algorithm are highlighted. In case of ties, the one using the fewest number of input features is considered optimized.

Algorithm	P_{acc} top 5 features	P_{acc} top 4 features	P_{acc} top 3 features	P_{acc} top 2 features	P_{acc} top 1 features
Logistic regression	0.21	0.21	0.21	0.21	**0.21**
kNN	**0.45**	0.42	0.42	0.34	0.34
CART	**0.87**	0.72	0.47	0.34	0.34
Naïve bayes	**0.27**	0.26	0.26	0.25	0.25
MLP	0.35	**0.38**	0.37	0.34	0.21
Random forest	**0.85**	0.71	0.47	0.34	0.34

The results from table 5.9 show that kNN, CART, Naïve bayes and Random forest perform best when using the top five features, whereas MLP and Logistic regression have optimized performances when using the top four and top one features respectively.

5.3.3 Optimizing hyperparameters

Table 5.10 – Shows the effect of optimizing the hyperparameters of kNN and MLP using their top input features

Algorithm	Default hyperparameter setting	Optimized settings	Default performance P_{acc}	optimized performance (accuracy, $\overline{F1}_{test}$, $\overline{F1}_{diff}$)	optimized performance P_{acc}
kNN	$k = 5$	$k = 1$	0.45	(0.81, 0.40, -0.47)	0.87
MLP	n.o. hidden nodes: 100	n.o. hidden nodes: 100	0.35	(0.85, 0.33, -0.01)	0.34

As shown in table 5.10, the optimized settings for MLP is the same as its default settings. Its overall performance was slightly reduced in comparison to the results in table 5.8 and 5.9. This is believed to be due to non-deterministic behavior of MLP. Table 5.10 shows that kNN was significantly improved using $k = 1$. Based on its new accumulated performance score: $P_{acc} = 0.87$, it tied with CART as the top performing algorithm as seen in table 5.9.

5.3.4 Results and analysis

Table 5.11 – Shows the overall optimized settings and performances for each of the algorithms in classifiying precipitation type. The best performing algorithms are highlighted.

Algorithm	Optimized settings	Best choice of input features	Best performance (Accuracy, $\overline{F1}_{test}$, $\overline{F1}_{diff}$)	Best performance P_{acc}
Logistic regression	Scikit default	top 1	(0.81, 0.21, 0.00)	0.21
kNN	**k = 1**	**top 5**	**(0.81, 0.40, -0.47)**	**0.87**
CART	**Scikit default**	**top 5**	**(0.84, 0.46, -0.41)**	**0.87**
Naïve bayes	Scikit default	top 5	(0.82, 0.27, 0.00)	0.27
MLP	Scikit default	top 4	(0.85, 0.38, 0.00)	0.38
Random forest	Scikit default	top 5	(0.85, 0.45, -0.40)	0.85

Table 5.11 shows that kNN and CART are tied for highest P_{acc} score among all algorithms. To break the tie, CART is chosen over kNN since CART has a higher accuracy- and $\overline{F1}$ score. CART is thus the algorithm that performs best in terms of performance and generalization when it comes to classifying precipitation type. CART has a performance score of $\overline{F1}_{test} = 0.46$ and accuracy = 0.82. This means that out of all observations in the test dataset, CART correctly classified 82% of the precipitation types correctly in the test dataset.

However, $\overline{F1}_{test} = 0.46$ indicates that the accuracy score is misleading. The recall scores in table 5.12 reveal that CART performed well in classifying no precipitation: 90%, and that lower scores were obtained with classifying the other precipitation types. The lowest recall score is classifying rain and snow mixed. Rain and snow mixed was correctly identified in 3% of the cases when rain and snow mixed appeared in the test dataset.

Table 5.12 – Shows how CART performs in classifying each of the different precipitation types using the optimized setup as shown in table 5.11

Value	Precipitation type	Precision	Recall	F1	N.o. occurrences
1	no precipitation	0.90	0.92	0.91	18912
2	rain with >= 0°C air temperature	0.55	0.47	0.51	3222
3	rain with < 0°C air temperature	0.26	0.28	0.27	50
4	snow	0.61	0.55	0.58	781
6	rain and snow mixed	0.03	0.03	0.03	71

5.4 Predicting precipitation amount (Optic Eye)

5.4.1 Input features correlation ranking

Table 5.13 – Input features correlation ranking to the target feature: precipitation amount.

Relevancy ranking	Input feature	Correlation score
1	road friction	4478.75
2	road surface condition	2793.48
3	DST111 road surface temperature	234.64
4	month	60.36
5	hour	19.93

Table 5.13 shows that the road friction and road surface condition have high correlation to precipitation amount while the time-related features are less relevant.

5.4.2 Spot-checking

Table 5.14 – Results from spot-checking experiment on the top features for predicting precipitation amount. The results are shown as a tuple: (MSE_{test}, MSE_{diff}) where MSE_{test} represents performance and $MSE_{diff} = MSE_{test} - MSE_{train}$ shows the degree of overfitting, larger values of MSE_{diff} indicate overfitting.

Algorithm	top 5 features	top 4 features	top 3 features	top 2 features	top 1 features
OLS	(0.58, -0.06)	(0.58, -0.06)	(0.58, -0.06)	(0.58, -0.06)	(0.58, -0.06)
CART	(1.01, 0.94)	(0.63, 0.18)	(0.98, 0.91)	(0.98, 0.80)	(0.62, 0.17)
kNN	(0.60, 0.15)	(0.61, 0.07)	(0.58, 0.15)	(0.62, 0.15)	(0.63, 0.05)
MLP	(0.56, -0.06)	(0.57, -0.05)	(0.55, -0.05)	(0.56, -0.06)	(0.55, -0.06)
Lasso	(0.61, -0.05)	(0.61, -0.05)	(0.61, -0.05)	(0.61, -0.05)	(0.61, -0.05)
Random forest	(0.67, 0.52)	(0.59, 0.13)	(0.68, 0.51)	(0.71, 0.46)	(0.57, 0.11)

Table 5.15 – Shows an accumalated performance score $P_{acc} = MSE_{test} + MSE_{diff}$. Top results for each algorithm are highlighted. In case of ties, the one with fewest input features is considered optimized.

Algorithm	P_{acc} top 5 features	P_{acc} top 4 features	P_{acc} top 3 features	P_{acc} top 2 features	P_{acc} top 1 features
OLS	0.52	0.52	0.52	0.52	**0.52**
CART	1.95	0.81	1.89	1.78	**0.79**
kNN	0.75	0.68	0.73	0.77	**0.68**
MLP	0.50	0.52	0.50	0.50	**0.49**
Lasso	0.56	0.56	0.56	0.56	**0.56**
Random forest	1.19	0.72	1.19	1.17	**0.68**

The results from table 5.15 indicate that all algorithms performs at best when the top one feature is used alone.

5.4.3 Optimizing hyperparameters

Table 5.16 – Shows the effect of optimizing the hyperparameters of kNN, MLP and Lasso using their top input features

Algorithm	Default hyperparameter setting	Optimized settings	Default performance P_{acc}	optimized performance (MSE_{test},MSE_{diff})	optimized performance P_{acc}
kNN	$k = 5$	$k = 64$	0.68	(0.54, -0.05)	0.49
MLP	n.o. hidden nodes: 100	n.o. hidden nodes: 64	0.49	(0.55, -0.06)	0.49
Lasso	$\lambda = 1$	$\lambda = 0.001$	0.56	(0.58, -0.06)	0.52

Table 5.16 shows that the results of kNN and Lasso were improved using the optimized hyperparameter settings while MLP produced the same result.

5.4.4 Results and analysis

Table 5.17 – Shows the overall optimized settings and performances for each of the algorithms in predicting precipitation amount. The best performing algorithms are highlighted.

Algorithm	Optimized settings	Best choice of input features	Best performance (MSE_{test}, MSE_{diff})	Best performance P_{acc}
OLS	Scikit default	top 1	(0.58, -0.06)	0.52
CART	Scikit default	top 1	(0.62, 0.17)	0.79
kNN	**k = 64**	**top 1**	**(0.54, -0.05)**	**0.49**
MLP	**n.o. hidden nodes: 256**	**top 1**	**(0.55, -0.06)**	**0.49**
Lasso	$\lambda = 0.001$	top 1	(0.58, -0.06)	0.52
Random forest	Scikit default	top 1	(0.57, 0.11)	0,68

As is shown in table 5.17, MLP and kNN are tied for the lowest P_{acc} score among all algorithms. To break the tie, kNN is considered a better choice by the author since kNN with $k = 64$ is believed to be less complex than MLP with 256 hidden nodes. This means that kNN is the algorithm that performs best in terms of performance and generalization in predicting precipitation amount. Its top performance: $MSE_{test} = 0.54$ means that in predicting precipitation amount, kNN was on average off by $\sqrt{0.54} \approx 0.73$ mm/30 min from the actual precipitation amount observations in the test dataset. Figure 5.2 shows the relationship between predictions made by MLP on the test dataset and actual values.

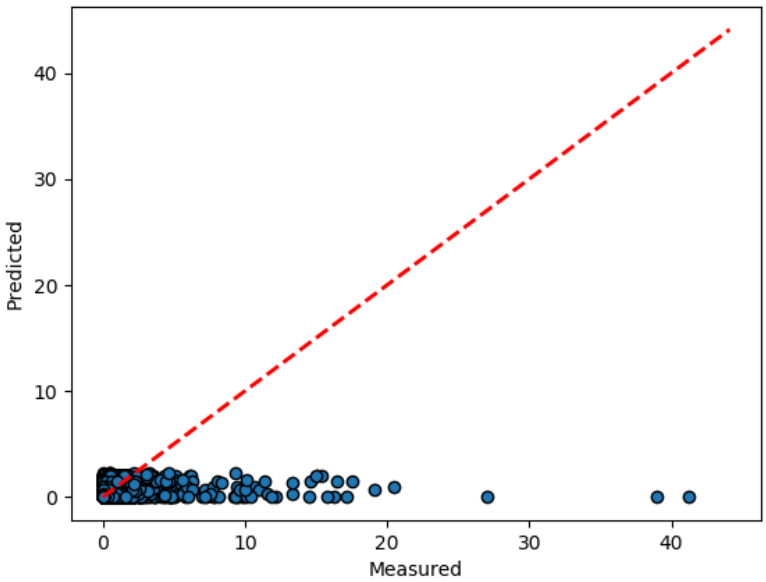

Figure 5.2 – Shows the relationship between actual measurements and predictions made by kNN to predict precipitation amount. The closer a point is to the red dashed line, the more accurate prediction for that observation.

Figure 5.2 shows a substantial difference between some measurements and predictions. Two measurements near 40 mm/30 min was predicted to be approximately 0 mm/30 min. The author suspected that some of the higher measurements of precipitation amount in the dataset was due to errors rather than being abnormal, but correct, measurements. The author consulted with Jonas Hallenberg at Trafikverket to see if some of the observations from the training dataset should be removed. Although he could not declare a definitive limit for a reasonable prediction amount, he recommended the author to set a limit at 20 mm/30 min [69].

The author decided to test an engineered version of the dataset in which all observations where precipitation amount is above 20 mm/30 min were removed, and run kNN again with the same settings as seen in table 5.17. A total of 27 observations were removed, resulting in performance score $MSE_{newtest} = 0.31$ which is an improvement from $MSE_{test} = 0.54$. Although this experiment was conducted outside the frame of the experimental methodology of this project, it was decided to use its results as the new highest performance score: $MSE_{test} = 0.31$. Using the new results means that kNN was on average off by $\sqrt{0.31} \approx 0.56$ mm/30 min from the actual measurements in the test dataset.

5.5 Predicting road surface temperature (DST111)

5.5.1 Input features correlation ranking

Table 5.18 – Relevancy of each possible input features to the target feature: DST111 road surface temperature.

Relevancy ranking	Input feature	Correlation score
1	road surface condition	11636.00
2	road friction	7411.80
3	month	4990.95
4	precipitation type	1897.37
5	hour	1784.49
6	precipitation amount	234,64

The correlation ranking from table 5.18 shows that road surface condition and road friction are the most relevant input features.

5.5.2 Spot-checking

Table 5.19 – Results from spot-checking experiment on the top features for predicting DST111 road surface temperature. The results are shown as a tuple: (MSE_{test}, MSE_{diff}) where MSE_{test} represents performance and $MSE_{diff} = MSE_{test} - MSE_{train}$ shows the degree of overfitting, larger values of MSE_{diff} indicate overfitting.

Algorithm	top 6 features	top 5 features	top 4 features	top 3 features	top 2 features	top 1 features
OLS	(70.45, 0.20)	(70.50, 0.22)	(71.69, 0.21)	(71.70, 0.21)	(75.25, 0.06)	(75.64, 0.12)
CART	(10.47, 1.54)	(10.27, 1.03)	(20.06, 0.50)	(20.56, 0.48)	(60.20, 0.01)	(60.93, -0.12)
kNN	(11.83, 0.65)	(11.90, 0.82)	(22.60, 0.44)	(23.36, 0.34)	(62.68, -0.22)	(61.66, -0.13)
MLP	(11.46, 0.27)	(13.14, 0.13)	(22.33, 0.34)	(22.44, 0.30)	(61.00, -0.15)	(62.49, -0.20)
Lasso	(71.43, 0.19)	(71.43, 0.19)	(72.65, 0.20)	(72.65, 0.20)	(76.38, 0.04)	(76.38, 0.04)
Random forest	(10.16, 1.11)	(10.16, 0.86)	(20.04, 0.46)	(20.56, 0.46)	(60.20, 0.00)	(60.93, -0.12)

Table 5.20 – Shows an accumalated performance score $P_{acc} = MSE_{test} + MSE_{diff}$. Top results for each algorithm are highlighted. In case of ties, the one using the fewest amount of input features is considered optimized.

Algorithm	P_{acc} top 6 features	P_{acc} top 5 features	P_{acc} top 4 features	P_{acc} top 3 features	P_{acc} top 2 features	P_{acc} top 1 features
OLS	**70.65**	70.72	71.90	71.91	75.31	75.52
CART	12.01	**11.30**	20.56	21.04	60.21	60.81
kNN	**12.48**	12.72	23.04	23.70	62.46	61.53
MLP	**11.73**	13.27	22.67	22.74	60.85	62.29
Lasso	**71.62**	71.63	72.85	72.85	76.42	76.42
Random forest	11.27	**11.02**	20.50	21.02	60.20	60.81

The results from table 5.20 show that OLS, kNN, MLP and Lasso performs at best when all top six features are used, while CART and Random forest benefit most from using the top five features.

5.5.3 Optimizing hyperparameters

Table 5.21 – Shows the effect of optimizing the hyperparameters of kNN, MLP and Lasso using their top input features

Algorithm	Default hyperparameter setting	Optimized settings	Default performance P_{acc}	optimized performance (MSE_{test},MSE_{diff})	optimized performance P_{acc}
kNN	$k = 5$	$k = 32$	12.48	(10.69, 0.43)	11.12
MLP	n.o. hidden nodes: 100	n.o. hidden nodes: 64	11.73	(11.19, 0.29)	11.48
Lasso	$\lambda = 1$	$\lambda = 0.001$	71.62	(70.46, 0.21)	70.67

All three algorithms attain a slightly better P_{acc} score in using their optimized hyperparameter settings.

5.5.4 Results and analysis

Table 5.22 – Shows the overall optimized settings and performances for each of the algorithms in predicting DST111 road surface temperature. The best performing algorithm is highlighted.

Algorithm	Optimized settings	Best choice of input features	Best performance (MSE_{test}, MSE_{diff})	Best performance P_{acc}
OLS	Scikit default	top 6	(70.45, 0.20)	70.65
CART	Scikit default	top 5	(10.27, 1.03)	11.30
kNN	$k = 32$	top 6	(11.83, 0.65)	12.48
MLP	n.o. hidden nodes: 256	top 6	(11.46, 0.27)	11.73
Lasso	$\lambda = 0.001$	top 6	(71.43, 0.19)	71.62
Random forest	**Scikit default**	**top 5**	**(10.16, 0.86)**	**11.02**

From table 5.22 it is revealed that Random forest has the lowest P_{acc} score and is thus the best algorithm in predicting DST111 road surface temperature. The top performance score for random forest is $MSE_{test} = 10.16$. This means that in predicting DST111 road surface temperature, the predictions made by Random forest was on average off by $\sqrt{10.16} \approx 3.19°$C from the actual temperature measurements in the test dataset. Figure 5.3 shows the relationship between predictions made by MLP on the test dataset and actual values.

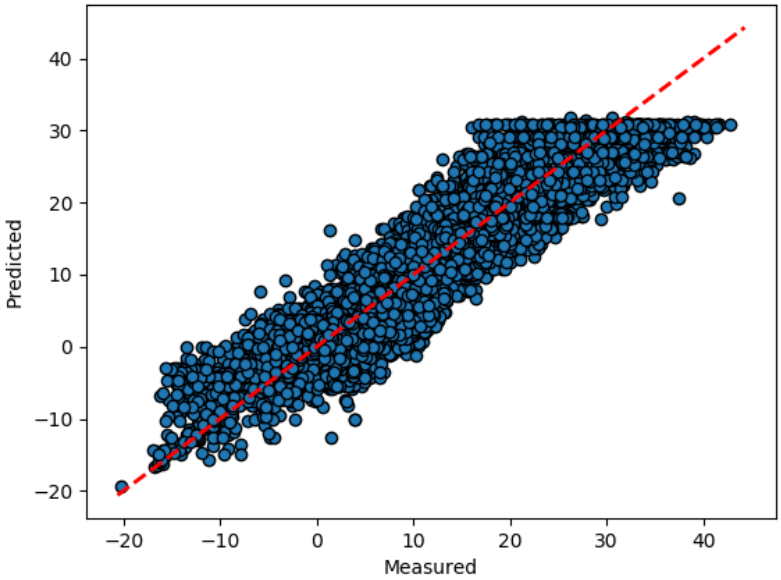

Figure 5.3 – Shows the relationship between actual measurements and predictions made by Random forest to predict DST111 road surface temperature. The closer a point is to the red dashed line, the more accurate prediction for that observation.

Although DST111 and TIRS both measure road surface temperature, the optimal performances of the two have a substantial difference. Random forest was the best at predicting for the DST111 with a performance score of $MSE_{test} = 10.16$, and MLP was the best performer in predicting TIRS road surface temperature with a score of $MSE_{test} = 0.88$. One difference between the two tasks is that DST111 road surface temperature can be used as input feature to predict TIRS road surface temperature, but not the other way around (see thesis objective in 1.2). It was decided to test the importance of having road surface temperature as input feature to predict another road surface temperature. The results obtained were not used to solve the thesis subtasks, but rather to gain an understanding of why the two performance results differed by a large degree.

Two holdout regression spot-checking experiments were carried out to see the importance of having either TIRS- or DST111 road surface temperature to predict the other. The first experiment is to predict TIRS road surface temperature but to not use its top relevant input feature: DST111 road surface temperature. The second experiment is to predict DST111 road surface temperature allowing TIRS road surface temperature to be used as input feature. In this project, top performance for predicting TIRS road surface temperature was $MSE_{test} = 0.88$, this was when DST111 was allowed as input feature. When not using DST111 as input feature, the best performance among algorithms to predict TIRS road surface temperature was Random forest with a reduced performance

score of $MSE_{test} = 15.36$. In the case of predicting DST111 road surface temperature, it was not allowed to use TIRS road surface temperature as input feature in this project, resulting in a performance score of $MSE_{test} = 10.16$. In allowing TIRS to be used as input feature in this experiment, the best performing algorithm to predict DST111 road surface temperature was MLP with a performance score of $MSE_{test} = 0.77$.

Chapter 6

Summary

This project has dealt with investigating whether or not RWIS sensors can be modelled using supervised machine learning algorithms. The sensors that are of interest to model are as follows:

- Optic Eye: Measures precipitation type and precipitation amount.
- TIRS: Dug into the road. Measures road surface temperature.
- DST111: Measures road surface temperature via infrared laser.

The idea was to model the behavior of the sensors listed above from data provided by the other sensors, except for the TIRS since it may be removed in the future. In addition to the sensors listed above, any algorithm model may train on data provided by the following sensors/instruments as well:

- DSC111: Measures road surface condition and road friction.
- MS4: Timestamp (this was made into two input features: hour and month in 4.3).

The objective of the project as seen in section 1.2 is as follows:

"The objective of this thesis is to find optimized supervised learning algorithms, in terms of performance and generalization, which model the behavior of the following sensors: Optic Eye, TIRS and DST111. Each sensor is modelled using measurements made by the other sensors, except for TIRS. Any algorithm may train on the following input features as well: timestamp, road friction and road surface condition. Timestamp is provided by the RWIS computer MS4. Road friction and road surface condition are measured by a sensor called DSC111."

The objective was broken down into four subtasks. Solutions to each subtask are covered in the sections that follow.

6.1 Classifying precipitation type

It was found in the results presented in section 5.3.4 that the best performing algorithm in terms of both performance and generalization in classifying precipitation type was CART

on Scikit-learn default settings, which attained a performance score of $\overline{F1}_{test} = 0.46$ and accuracy = 0.84. The following input features were used to obtain the optimized performance: road surface condition, road friction, DST111 road surface temperature, hour and month. Both kNN and CART tied for best accumulated performance scores $P_{acc} = 0.87$, but CART was chosen over kNN due to it having a higher accuracy- and $\overline{F1}$ score.

6.2 Predicting precipitation amount

The results from section 5.4.4 show that the best algorithm in terms of performance and generalization in predicting precipitation amount is kNN with $k = 64$. This resulted in a performance score of $MSE_{test} = 0.31$. Road friction was the only input feature used to obtain the optimized performance score. MLP and kNN tied for best accumulated performance scores $P_{acc} = 0.49$, but kNN was chosen over MLP based on a hypothesis that kNN is less complex.

6.3 Predicting TIRS road surface temperature

It was found in section 5.2.4 that the best algorithm in terms of performance and generalization for predicting TIRS road surface temperature is MLP with 64 hidden nodes. This gave a performance score of $MSE_{test} = 0.88$. The following input features were used to find the optimized performance: DST111 road surface temperature, road surface condition, road friction, month, hour and precipitation type. MLP was identified as the top performing algorithm based on it having the best accumulated performance score $P_{acc} = 0.89$.

6.4 Predicting DST111 road surface temperature

From the results in section 5.5.4, it was found that Random forest on Scikit-learn default settings was the top performing algorithm in terms of performance and generalization. It obtained a performance score of $MSE_{test} = 10.16$. The following input features were used to obtain the optimized performance: road surface condition, road friction, month, precipitation type and hour. Random forest was identified as the top performing algorithm based on it having the best accumulated performance score $P_{acc} = 11.02$.

Chapter 7

Discussion and conclusions

This chapter presents a dicsussion on the results obtained, and the methods used. It is followed up with conclusions connected to the aim of this project and ultimately recommendations for Trafikverket.

7.1 Discussion

The first part of this section covers a discussion on the results obtained, and the methods used to obtain the results are discussed in the second part.

7.1.1 Results

It is up to Trafikverket to decide what a reasonable margin for error is in stating that a model can or cannot model the behavior of a sensor, but when comparing the results from the regression subtasks in this project, modelling DST111 road surface temperature shows a higher error rate than that of precipitation amount and TIRS road surface temperature. The results from the experiments in section 5.5.4 reveals the importance of using data from a different road surface temperature sensor a as input feature when modelling a road surface temperature sensor b. The results indicate that the modelling of either of the two sensors (DST111 or TIRS) is improved when doing so. This suggests that using a strongly linearly correlated input feature, such as a different road surface temperature sensor, improves the model significantly. Given that TIRS road surface temperature was not allowed to be used as input feature to predict DST111 road surface temperature, alternative input features may be needed to build a decent model. Road surface temperature is not the only type of temperature the RWIS measure, they also measure air temperature. The author speculates that air temperature have a linear correlation to road surface temperature, and could therefore likely improve the results of modelling DST111 road surface temperature if it is used as input feature.

As for the classification task of classifying precipitation type, the author deems that its performance should be evaluated on its $\overline{F1}$ score rather than on its accuracy score. The recall scores from table 5.12 show that the best model for modelling precipitation

type performed well in predicting no precipitation: 90% while the other precipitation types had significantly lower recall scores, the lowest being rain and snow mixed which was correctly classified in 3% of its occurrences. The relatively low $\overline{F1}$ score may be due to the fact that the dataset is imbalanced in terms of precipitation type, and that the imbalanced data problem was not successfully handled in this project.

In this project, optimization of hyperparameters was done based on an algorithm's performance on the test dataset. This means that information from the test dataset was not necessarily previously unseen for any algorithm involved. If an independent validation dataset was used instead to optimize hyperparameters, information would not "leak" from the test dataset in order to achieve a high performance. The author recommend readers to take note of this fact when reviewing the results of this project.

7.1.2 Method

The method for calculating an accumulated performance score presented in section 3.9.2 could possibly be improved by not allowing negative values in the overfitting score. An overfitting score of zero means that the model performs equally well on both the training and test dataset. A negative overfitting score means a model performed better on the test dataset than on its training dataset. The author thinks that a negative overfitting score is not necessarily better than one close to zero. A model with a negative overfitting score increases its accumulated performance score. This may result in a non-optimal choice of algorithm having the best accumulated performance score, and thus being identified as the top performer for a given problem.

In this project, input features were ranked in relevancy to a given target feature based on the degree of linear correlation. Non-linear relationships between features, that may or may not present a small degree of linear relationship, did not have as big an impact as a strong linear relationship has on the feature ranking. If there are non-linear relationships among features, feature ranking based primarily on linear correlations may not have been optimal. One way could be to test all possible combinations of features instead of relying on an input feature ranking. However, that would require much more time than varying the amount of top relevant input features. Equation 7.1 shows how many combinations there are with six input features, which is the same amount of input features used to predict TIRS road surface temperature.

$$c(n = 6) = \sum_{r=1}^{6} \frac{6!}{r!(6 - r)!} = 63 \tag{7.1}$$

The author thinks that feature ranking was suitable for this project since it's much less time consuming than to test dozens of combinations of input features for each subtask.

In this project it was decided to combine the data provided by the six weather stations along state road E6 in Sweden. The distance between the two stations that are furthest away from each other is approximately 140 kilometers [70]. An alternative approach could be to keep the data separated in six datasets, one per RWIS, and find optimized models for each RWIS. The author hypothesize that this could work if data provided by

each station is big enough for training, and if differences in weather conditions among the weather stations are significant, so that it is beneficial to train models from local data alone. Another approach could be to keep the data combined, but to add a feature that shows which RWIS an observation is taken from. This could capture local weather condition patterns, while being able to learn from other stations as well, which could prove useful. For example, in the case of abnormal weather conditions for one station S_1, such weather conditions might be frequently occurring for another station S_2, in which case S_1 can learn from S_2. This approach could also potentially benefit from having a larger dataset overall, and to avoid the effort in training models separately.

7.2 Conclusions

The aim of this project was to find optimized supervised learning algorithms to model the behavior of three sensors: Optic Eye, Track Ice Road Sensor (TIRS) and DST111. This was desired so that Trafikverket can potentially replace existing sensors to reduce costs, or use as backup in case of failing sensors. The sensors make four different type of measurements in total: precipitation type, precipitation amount, TIRS road surface temperature and DST111 road surface temperature. The objective was broken down into four subtasks which aimed at finding optimized algorithms to model each of the measurements made by the sensors.

The results obtained in this project indicate that the measurements made by Optic Eye: precipitation type and precipitation amount, are best modelled using Classification And Regression Tree (CART) for precipitation type and k-nearest neighbor (kNN) for precipitation amount. An accuracy score of 0.84 and $\overline{F1}$ score of 0.46 was obtained in modelling precipitation type using Scikit-learn default settings, and the following input features: road surface condition, road friction, DST111 road surface temperature, hour and month. The $\overline{F1}$ score in classifying precipitation type can likely be improved by addressing its imbalanced dataset representation. Precipitation amount was best modelled using kNN, obtaining a performance score $MSE = 0.31$ with $k = 64$ using road friction as the only input feature.

The two remaining sensors measuring road surface temperature: TIRS and DST111, were best modelled using Multi-Layer Perceptron (MLP) and Random forest respectively. MLP was set to use 64 hidden nodes, with which a performance score $MSE = 0.88$ was obtained in modelling TIRS road surface temperature using the following input features: DST111 road surface temperature, road surface condition, road friction, month, hour and precipitation type. As for modelling DST111 road surface temperature, the best model was obtained from using Random forest on Scikit-learn default settings with a performance score of $MSE = 10.16$ using the following input features: road surface condition, road friction, month, precipitation type and hour. The success of predicting either TIRS- or DST111 road surface temperature relies on having at least one strongly linearly correlated input feature, for instance: a different road surface temperature feature.

Apart from finding answers to the project subtasks, the author made some findings

along the way:

- The accumulated performance score used in this project to evaluate algorithms, in terms of test performance and generalization, is not an optimal evaluation score. It can be improved by not allowing negative overfitting scores in equation 3.4.
- If there are non-linear correlations among the different features, the overall results of this project can possibly be improved by testing all possible combinations of input features when modelling a given target feature, instead of using an input feature ranking method where linear correlations have the biggest impact. However, such testing requires more time, and as of such the author recommends to test all input features on low-dimensional datasets, and to consider input feature ranking for high-dimensional datasets. Any practicer considering input feature ranking should also consider if the ranking should focus on the degree of linear correlation, or some other type of measurement.

7.3 Recommendations

Even though it is up to Trafikverket to decide when a model performs sufficientlly well in modelling a target feature, the author recommends Trafikverket to refrain from modelling DST111 road surface temperature using the given algorithm, algorithm settings and input features since it has relatively bad performance when compared to the performance of modelling TIRS road surface temperature. Trafikverket should investigate the possibility of modelling TIRS road surface temperature and precipitation amount using the algorithms and algorithm settings seen in table 5.1. However, Trafikverket should experiment with other input features from the RWIS as well, especially air temperature could potentially improve the performance of modelling DST111 road surface temperature.

Given that all precipitation types are equally important to classify, the author recommends Trafikverket to investigate if the imbalanced data problem of precipitation type can be solved. This can be solved either by collecting more data from the less represented classes, or to use similar techniques as the ones used in this project (see attempts in handling class imbalance in section 4.3.2).

The dataset used in this project is limited to a region in western Sweden along state road E6. The author recommends Trafikverket to investigate if models can benefit from training on data from as many RWIS as possible, or if training with data from stations that are geographically close proves a better choice. If the observations from several RWIS are combined to one dataset, Trafikverket should experiment with having a feature stating which RWIS the observation is taken from.

Bibliography

[1] A. Arvidsson, "The Winter Model – A new way to calculate socio-economic costs depending on winter maintenance strategy", *Cold Regions Science and Technology Volume 136, April 2017, Pages 30-36*, 2017.

[2] Trafikverket, *Trafikverkets årsredovisning 2015*, 2016.

[3] Trafikverket. (2017). Vintervägh̊allning, [Online]. Available: `https://www.trafikverket.se/resa-och-trafik/underhall-av-vag-och-jarnvag/Sa-skoter-vi-vagar/Vintervaghallning/` (visited on 02/07/2018).

[4] Pelpet. (2010). Rwis Station at sensor site Myggsjön, [Online]. Available: `https://commons.wikimedia.org/wiki/File:Rwis_station_Myggsjon_01.JPG` (visited on 02/06/2018).

[5] V. Moberg, private communication, 2018.

[6] Trafikverket, *RFI RWIS asked questions*, 2017.

[7] Trafikverket, *Road weather information systems*, n.d.

[8] Trafikverket, *Nästa generation VägVäderinformationsSystem (VViS)*, n.d.

[9] Vaisala. (n.d.). DST111 Remote Surface Temperature Sensor, [Online]. Available: `https://www.vaisala.com/en/products/instruments-sensors-and-other-measurement-devices/weather-stations-and-sensors/dst111` (visited on 04/17/2018).

[10] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[12] S. Gollapudi, *Practical Machine Learning*, ser. Community experience distilled. Packt Publishing, 2016, ISBN: 9781784399689. [Online]. Available: `https://books.google.se/books?id=3ywhjwEACAAJ`.

[13] D. Chicco, "Ten quick tips for machine learning in computational biology.", *BioData Mining*, vol. 10, pp. 1 –17, 2017, ISSN: 17560381. [Online]. Available: `http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aph&AN=126676440&lang=sv&site=eds-live&scope=site`.

[14] A. Maxwell, L. Runzhi, Y. Bei, W. Heng, O. Aihua, H. Huixiao, Z. Zhaoxian, G. Ping, and Z. Chaoyang, "Deep learning architectures for multi-label classification of intelligent health risk prediction.", *BMC Bioinformatics*, vol. 18, pp. 121 –131, 2017, ISSN: 14712105. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aph&AN=127122779&lang=sv&site=eds-live&scope=site.

[15] Scikit-learn. (n.d.). train test split, [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (visited on 05/12/2018).

[16] F. Lardinois. (2017). Google says its machine learning tech now blocks 99.9% of Gmail spam and phishing messages, [Online]. Available: https://techcrunch.com/2017/05/31/google-says-its-machine-learning-tech-now-blocks-99-9-of-gmail-spam-and-phishing-messages/ (visited on 03/19/2018).

[17] M. Aly, *Survey on Multiclass Classification Methods*, 2005.

[18] H. He and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*, 1st. Wiley-IEEE Press, 2013, ISBN: 1118074629, 9781118074626.

[19] Scikit-learn. (n.d.). From binary to multiclass and multilabel, [Online]. Available: http://scikit-learn.org/stable/modules/model_evaluation.html#from-binary-to-multiclass-and-multilabel (visited on 04/25/2018).

[20] T. Beysolow II, *Introduction to Deep Learning Using R: A Step-by-Step Guide to Learning and Implementing Deep Learning Models Using R*, 1st. Berkely, CA, USA: Apress, 2017, ISBN: 1484227336, 9781484227336.

[21] "Submodel Selection and Evaluation in Regression - the X-Random case.", *International Statistical Review*, vol. 60, no. 3, pp. 291 –319, n.d. ISSN: 03067734. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edswsc&AN=A1992KC62300004&lang=sv&site=eds-live&scope=site.

[22] S. Kotsiantis, "Supervised machine learning: A review of classification techniques.", *Informatica (Ljubljana)*, vol. 31, no. 3, pp. 249–268, 2007, ISSN: 03505596. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-36749047332&lang=sv&site=eds-live&scope=site.

[23] L. Hyafil and R. Rivest, "Constructing optimal binary decision trees is NP-complete.", *Information Processing Letters*, vol. 5, no. 1, pp. 15–17, 1976, ISSN: 00200190. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-0001815269&lang=sv&site=eds-live&scope=site.

[24] N. Lavrač, *Machine Learning: ECML 2003: 14th European Conference on Machine Learning, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings*, ser. Lecture Notes in Artificial Intelligence v. 14. Springer, 2003, ISBN: 9783540201212. [Online]. Available: https://books.google.se/books?id=C_E4lDNIR1QC.

[25] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997, ISBN: 0070428077, 9780070428072.

[26] scikit-learn developers. (n.d.). Decision trees, [Online]. Available: http://scikit-learn.org/stable/modules/tree.html (visited on 04/10/2018).

[27] H. Almuallim, "An efficient algorithm for optimal pruning of decision trees", *Artificial Intelligence*, vol. 83, no. 2, pp. 347 –362, 1996, ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(95)00060-7. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370295000607.

[28] scikit-learn developers. (n.d.). API Reference, [Online]. Available: http://scikit-learn.org/stable/modules/classes.html (visited on 04/10/2018).

[29] scikit-learn developers. (n.d.). Multiclass and multilabel algorithms, [Online]. Available: http://scikit-learn.org/stable/modules/multiclass.html (visited on 04/10/2018).

[30] X. Wu, V. Kumar, M. Steinbach, Q. Ross, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, A. Ng, B. Liu, P. Yu, Z.-H. Zhou, D. Hand, and D. Steinberg, "Top 10 algorithms in data mining.", *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008, ISSN: 02191377. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-37549018049&lang=sv&site=eds-live&scope=site.

[31] L. Zhong, L. Lin, Z. Lu, Y. Wu, Z. Lu, M. Huang, W. Yang, and Q. Feng, "Predict CT image from MRI data using KNN-regression with learned local descriptors", in *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, 2016, pp. 743–746. DOI: 10.1109/ISBI.2016.7493373.

[32] S. Okamoto and N. Yugami, "Effects of domain characteristics on instance-based learning algorithms", *Theoretical Computer Science*, vol. 298, no. 1, pp. 207 – 233, 2003, Selected Papers in honour of Setsuo Arikawa, ISSN: 0304-3975. DOI: https://doi.org/10.1016/S0304-3975(02)00424-3. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397502004243.

[33] P. Domingos and M. Pazzani, "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss", *Machine Learning*, vol. 29, no. 2, pp. 103–130, 1997, ISSN: 1573-0565. DOI: 10.1023/A:1007413511361. [Online]. Available: https://doi.org/10.1023/A:1007413511361.

[34] H. Zhang, "The Optimality of Naïve Bayes", in *In FLAIRS 2004 conference*, 2004.

[35] E. Frank, L. Trigg, G. Holmes, and I. H. Witten, "Technical Note: Naive Bayes for Regression", *Machine Learning*, vol. 41, no. 1, pp. 5–25, 2000, ISSN: 1573-0565. DOI: 10.1023/A:1007670802811. [Online]. Available: https://doi.org/10.1023/A:1007670802811.

[36] M. Aly, "Survey on multiclass classification methods", *Neural networks*, pp. 1–9, 2005.

[37] J. Heaton, *Introduction to Neural Networks with Java*. Heaton Research, 2008, ISBN: 9781604390087. [Online]. Available: https://books.google.se/books?id=Swlcw7M4uD8C.

[38] Scikit-learn. (n.d.). Multi-layer Perceptron, [Online]. Available: http://scikit-learn.org/stable/modules/neural_networks_supervised.html (visited on 05/12/2018).

[39] A. K. Chowdhury, D. Tjondronegoro, V. Chandran, and S. G. Trost, "Ensemble Methods for Classification of Physical Activities from Wrist Accelerometry.", *Medicine and Science in Sports and Exercise*, vol. 49, no. 9, pp. 1965 –1973, 2017, ISSN: 01959131. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=s3h&AN=124667245&lang=sv&site=eds-live&scope=site.

[40] R. Caruana and A. Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms", in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06, Pittsburgh, Pennsylvania, USA: ACM, 2006, pp. 161–168, ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143865. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143865.

[41] P. Domingos, "A Few Useful Things to Know About Machine Learning.", *Communications of the ACM*, vol. 55, no. 10, pp. 78 –87, 2012, ISSN: 00010782. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=82151052&lang=sv&site=eds-live&scope=site.

[42] J. Brownlee. (2017). Difference Between Classification and Regression in Machine Learning, [Online]. Available: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/ (visited on 03/22/2018).

[43] C. Dwork, T. Pitassi, V. Feldman, O. Reingold, M. Hardt, and A. Roth, "Generalization in adaptive data analysis and holdout reuse.", in *Advances in Neural Information Processing Systems*, vol. 2015-January, (1)Microsoft Research, 2015, pp. 2350–2358. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-84965181547&lang=sv&site=eds-live&scope=site.

[44] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Ser. Wadsworth Statistics/Probability Series. Wadsworth Advanced Books and Software, Belmont, CA, 1984, ISBN: 0-534-98053-8. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=msn&AN=MR726392&lang=sv&site=eds-live&scope=site.

[45] "No unbiased estimator of the variance of K-fold cross-validation.", *Journal of Machine Learning Research*, vol. 5, pp. 1089 –1105, n.d. ISSN: 15324435. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.

com/login.aspx?direct=true&db=edswsc&AN=000236328100001&lang=sv&site=eds-live&scope=site.

[46] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", Morgan Kaufmann, 1995, pp. 1137–1143.

[47] P. Gupta. (2017). Regularization in Machine Learning, [Online]. Available: https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a (visited on 04/03/2018).

[48] C. S. Göbl, L. Bozkurt, A. Tura, G. Pacini, A. Kautzky-Willer, and M. Mittlböck, "Application of Penalized Regression Techniques in Modelling Insulin Sensitivity by Correlated Metabolic Parameters.", *PLoS ONE*, vol. 10, no. 10, pp. 1 –19, 2015, ISSN: 19326203. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aph&AN=110795719&lang=sv&site=eds-live&scope=site.

[49] M. Bermingham, A. Spiliopoulou, C. Hayward, A. Wright, P. Navarro, Haley C.S. (1, R. Pong-Wong, I. Rudan, H. Campbell, J. Wilson, and F. Agakov, "Application of high-dimensional feature selection: Evaluation for genomic prediction in man.", *Scientific Reports*, vol. 5, 2015, ISSN: 20452322. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-84930221871&lang=sv&site=eds-live&scope=site.

[50] J. P. Kandhasamy and S. Balamurali, "Performance Analysis of Classifier Models to Predict Diabetes Mellitus", *Procedia Computer Science*, vol. 47, pp. 45 –51, 2015, Graph Algorithms, High Performance Implementations and Its Applications (ICGHIA 2014), ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.03.182. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915004500.

[51] Scikit-learn. (n.d.). Feature selection, [Online]. Available: http://scikit-learn.org/stable/modules/feature_selection.html (visited on 05/10/2018).

[52] J. Mueller and L. Massaron, *Machine Learning For Dummies*. Ser. For Dummies. For Dummies, 2016, ISBN: 9781119245513. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1237397&lang=sv&site=eds-live&scope=site.

[53] S. S. Patil and S. P. Sonavane, "Enriched Over_Sampling Techniques for Improving Classification of Imbalanced Big Data", in *2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService)*, 2017, pp. 1–10. DOI: 10.1109/BigDataService.2017.19.

[54] imbalanced learn. (n.d.). imbalanced-learn, [Online]. Available: https://github.com/scikit-learn-contrib/imbalanced-learn (visited on 04/25/2018).

[55] C. C. Aggarwal, *Data Mining: The Textbook*. Springer Publishing Company, Incorporated, 2015, ISBN: 3319141414, 9783319141411.

[56] imbalanced learn. (2016). imblearn.over_sampling.SMOTE, [Online]. Available: http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.over_sampling.SMOTE.html#imblearn.over_sampling.SMOTE (visited on 04/25/2018).

[57] J. Brownlee. (2014). A Gentle Introduction to Scikit-Learn: A Python Machine Learning Library, [Online]. Available: https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/ (visited on 05/01/2018).

[58] A. Abraham, A. Abraham, F. Pedregosa, M. Eickenberg, P. Gervais, A. Gramfort, B. Thirion, G. Varoquaux, A. Mueller, and J. Kossaifi, "Machine learning for neuroirnaging with scikit-learn.", *FRONTIERS IN NEUROINFORMATICS*, vol. 8, n.d. ISSN: 16625196. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edswsc&AN=000348105800001&lang=sv&site=eds-live&scope=site.

[59] S. Seo, J. Choi, S. K. Ahn, K. W. Kim, J. Kim, J. Choi, J. Kim, and J. Ahn, "Prediction of GPCR-Ligand Binding Using Machine Learning Algorithms.", *Computational & Mathematical Methods in Medicine*, pp. 1 –5, 2018, ISSN: 1748670X. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aph&AN=127664788&lang=sv&site=eds-live&scope=site.

[60] Microsoft. (2017). Cheat sheet: How to choose a MicrosoftML algorithm, [Online]. Available: https://docs.microsoft.com/en-us/machine-learning-server/r/how-to-choose-microsoftml-algorithms-cheatsheet (visited on 04/18/2018).

[61] Scikit-learn. (n.d.). Choosing the right estimator, [Online]. Available: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html (visited on 04/18/2018).

[62] J. Brownlee. (2014). Why you should be Spot-Checking Algorithms on your Machine Learning Problems, [Online]. Available: https://machinelearningmastery.com/why-you-should-be-spot-checking-algorithms-on-your-machine-learning-problems/ (visited on 05/01/2018).

[63] Scikit-learn. (n.d.). Lasso, [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html (visited on 05/12/2018).

[64] D. Pyle, *Data Preparation for Data Mining*, 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, ISBN: 1558605290, 9781558605299.

[65] M. Refaat, *Data Preparation for Data Mining Using SAS.* Ser. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2007, ISBN: 9780123735775. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=195005&lang=sv&site=eds-live&scope=site.

[66] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed., ser. Morgan Kaufmann Series in Data Management Systems. Amsterdam: Morgan Kaufmann, 2011, ISBN: 978-0-12-374856-0. [Online]. Available: http://www.sciencedirect.com/science/book/9780123748560.

[67] Scikit-learn. (n.d.). SelectKBest, [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html (visited on 05/09/2018).

[68] Scikit-learn. (n.d.). Univariate feature selection, [Online]. Available: http://scikit-learn.org/stable/modules/feature_selection.html\#univariate-feature-selection (visited on 05/09/2018).

[69] J. Hallenberg, private communication, 2018.

[70] Google. (n.d.), [Online]. Available: https://www.google.se/maps/dir/Surte/58.8788466,11.2751227/@58.8847352,11.231009,11.69z/data=!4m9!4m8!1m5!1m1!1s0x46455fbf9e7999a3:0xe74e7297e5224ddf!2m2!1d12.0157441!2d57.827921!1m0!3e0 (visited on 06/05/2018).