Using supervised learning algorithms to model the behavior of road weather-related sensors

Tobias Axelsson

9 maj 2018

Acknowledgements

I am a student blalsadf

Abstract

asdasd

Sammanfattning

asdasdasd

Contents

Chapter 1

Introduction

The chapter starts with a background describing why road condition monitoring is important and who Trafikverket are, how road condition data is collected today and why the technology behind it needs improvement. Later on, machine learning basics are explained and how it can be used in this project. An objective for the project is defined followed by its delimitations. Lastly, a thesis structure is presented to simplify navigation through different parts of the project.

1.1 Background

Living in cold areas of the world usually means 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] presented 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 why it's needed, it can still present a notable economic cost for the organization(s) involved. Trafikverket, the agency in charge of road 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 helps the contracters monitor road conditions with their so-called Road Weather Information System (RWIS) [3]. Trafikverket has

around 800 RWIS (see 1.1) distributed across state roads in Sweden which are used by contracters to carry out winter road maintenance work [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/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	measurement timestamp	date (mmddhhmm)	all

Table 1.1 shows some of the sensors the operational RWIS uses. The sensors are connected to a computer nearby computer called MS4, but Trafikverket aims to replace MS4 with a new generation of computers by 2021 [7]. The computer has limited capacity to handle current and future contracter needs, such as real-time image transfers, and electric components are hard to replace [7].

In a personal interview with Johan Casselgren at Luleå University of Technology, he mentions that it is interesting to investigate if certain sensors, especially the Track Ice Road Sensor, can be replaced along with the new generation of computers. Johan Casselgren says that the Track Ice Road Sensors are dug into the road, which may require the road 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 useful since it measures road temperature remotely using infrared laser [8]. In addition to the sensors listed in 1.1, many of the stations are also equipped with a camera [9]. The operational RWIS also measure air humidity, air temperature, max-wind, wind-average and more, but neither these additional features nor the road photos taken by the camera are considered in this project since Johan Casselgren expressed an interest in investigating the relationship between the features in 1.1. Moreover, adding additional features may introduce the curse of dimensionality as brought up in 1.1.2.

The author, Johan Casselgren and Niklas Karvonen, who is also from Luleå University of Technology, concluded over personal communication that machine learning models can most likely be used to model the behavior of the Track Ice Road Sensor, and consequently, refrain from installing it in the future. The author and Johan Casselgren also sees potential in using machine learning models as a backup system when sensors malfunction, and therefore see benefits of modelling other sensors from 1.1 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 analyzing 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 processes a labeled dataset while unsupervised learning attempts to make sense of unlabeled data. As can be seen in 1.4, the data provided by Trafikverket to perform this project is labeled data in the form of column headers in Microsoft Excel workbooks. If, for some reason, the column headers were to be removed, it would qualify as an unlabeled dataset. Given that a labeled dataset is provided in this project, it makes sense to consider supervised learning algorithms.

Dimensionality in machine learning refers to how many features are used as input to the algorithm. A one dimensional dataset could for example contain 171425 observations with one feature: precipitation type. A tempting brute-force approach may be to include every possible feature available, not only those brought up in 1.1, but also road photos, wind-speed etc. This however, may lead to something called the curse of dimensionality,

which basically means that more dimensions in the dataset introduces complexity and possibly an increased amount of erros in the model [12]. During a personal meeting, Johan Casselgren also recommended to focus on using the features listed in 1.1. However, over an email conversation with Jonas Hallenberg who works at Trafikverket, difficulties were expressed in trying to model the DSC111 sensor. He says the problem is that Trafikverket uses salt on all the of the roads 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 the road surface condition is icy, which could be measured by DSC111, it may in fact be wet in reality. Johan casselgren suggests that data from DSC111 can be used to model the other sensors. Data from the Track Ice Road Sensor road temperature is not to be used as input when modelling remaining sensors since, as previously covered, Trafikverket plans rid of this sensor in the future.

The forementioned definition of machine learning by [10] mentions a performance measure P. This measure can be used to evaluate supervised learning algorithm's abilities to model a feature in 1.1, see 2.1 for more information on performance measures. 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 observations of the target feature as a mathematical graph and attempts to build a model that best fits the observations in the graph. But the algorithm is not necessarily perfect by having a high performance score. If the model is built in such a way that it fits the provided data perfectly, which may have outliers etc, it may have difficulties in predicting new data. This condition is known as overfitting, the opposite is called underfitting, both of which are covered in detail in 2.2. So not only is an algorithm with high performance desirable, also one whose model generalizes well so that both underfitting and overfitting is avoided.

1.2 Objective

The objective is to find optimal supervised learning models, in terms of performance and generalization, which models the behavior of the following sensors: Optic Eye, Track Ice Road Sensor and DST111 where each sensor is trained from observations made from other the other sensors but its own. In addition to the forementioned sensors, except for Track Ice Road Sensor, any model may train on the following input features as well: measurement timestamp, road friction and road surface condition.

1. Find the algorithm among supervised learning algorithms, that can be used to classify **precipitation type**, whose performance and generalization score is best. The data may contain the following input features:
    - measurement timestamp
    - DST111 road surface temperature
    - road friction
    - road surface condition

2. Find the algorithm among supervised learning algorithms, that can be used to predict **precipitation amount**, whose performance and generalization score is best. The data may contain the following input features:
    - measurement timestamp
    - DST111 road surface temperature
    - road friction
    - road surface condition
3. Find the algorithm among supervised learning algorithms, that can be used to predict **Track Ice Road Sensor road surface temperature**, whose performance and generalization score is best. The data may contain the following input features:
    - measurement timestamp
    - precipitation type
    - precipitation amount
    - DST111 road surface temperature
    - road friction
    - road surface condition
4. Find the algorithm among supervised learning algorithms, that can be used to predict **DST111 road surface temperature**, whose performance and generalization score is best. The data may contain the following input features:
    - measurement 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 2.1.1 and 2.1.2)
2. The algorithm is available in Scikit-learn [13]
3. The algorithm belongs to one of the following algorithm families (see 2.3):

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

The performance of supervised algorithms can generally be improved by optimizing its hyperparameters (see 2.4). However, there can be many ways that a single algorithm can be configured. Since this project deals with comparing performances of several algorithms, exploring all combination of hyperparameters would take a significant amount of time both in setting up experiments, and in actual running-time. When it comes to optimizing hyperparameters, it was decided to focus on the choice of k in kNN, the number of hidden nodes in Backpropagation, and magnitude of regularization λ in Lasso (see 2.3 and 2.2.2).

Some of the sensors, such as the DSC111 (see 4.1), are malfunctioning frequently. To avoid any complex relationships between functioning and malfunctioning sensors, it is decided to model non-error behavior by using non-error input features only.

In addition to the time 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. such that 2015 and 2016 are relatively similar in that sense. Also since the measurements are from 2015-2016, it is assumed that any model built using year as 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 etc. By these assumptions, it was decided to not consider year as an input feature.

Support vector machine (SVM) algorithms could 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 test (see 3.5.1) on the iris dataset in Scikit-learn, that SVM have significantly longer running-time than the other algorithms in 3.1.

1.4 Provided data

A dataset containing 171425 measurements (observations) was provided by Trafikverket to carry out this project. The observations are from 2015-2016 from six RWIS along state road E6 in Sweden, where every station measures the features seen in 1.1 roughly 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 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 1.1, precipitation type and road surface condition have discrete values that are explained in the readme file.

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

Chapter 2

Literature Review

The chapter gives both general and specific information on theory used in this project. It starts of with a general description of supervised learning, followed generalization and why it is important. It follows up with some information on supervised learning algorithms, how algorithm performance can be improved and lastly how to prepare datasets for applied machine learning.

2.1 Supervised learning

In supervised learning, the algorithm receives a dataset of labeled observations which are used to build a mathematical 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 be used potentially update its parameters in order to improve performance [12].
- Test dataset: Gives an unbiased evaluation of the final model.

Skocik et al. [14] call this a lock box approach. According to [12], the proportions of the split is usually 60% training, 30% test and 10% evaluation while [15] 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 [16]. It is suggested by [15] to employ the lock box approach in any machine learning project.

Supervised learning can be thought of as having a teacher supervising the algorithm. 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 works: Suppose a training, validation 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, as seen in Eq. 2.1, and possibly updated when validated against the validation dataset. Suppose the supervised learning algorithm used is Ordinary least squares (see section 2.3.4) 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 tested with the test dataset to see how it performs on unseen data.

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 a new observation into one of k categories (classes), $k \geq 2$ [11]. The problem of classifying new 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 [17]. 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 [18]. Classifying precipation type, which is done in this project, is a multiclass classification problem since it has more than two classes.

Another example of a classification problem, one that may well be the first that machine learning novices encounter, is classification of the Iris flower dataset. The dataset consists of 50 observations with four features: length and width of the sepals and petals, in centimeters. Based on this information, the problem is to classify an observation into one of three classes: Setosa, Versicolour, Virginica [19]. How the classification is carried out 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 for the Iris dataset, but their performance in doing so may differ. Performance of classifiers are typically measured in terms of accuracy, which is the amount of correct predictions divided by the number of observations in the test dataset.

$$\text{accuracy} = \frac{\#\text{correct predictions total}}{\#\text{observations}} \tag{2.3}$$

A high accuracy score such as 90% may seem promising, but what if 90% of the test data is made up of Setosa alone? That means that the model correctly classified

all Setosa observations, but failed on all of the Versicolour and Virginica classifications. This is probably an indication of an imbalanced dataset, which means that the different classes in the dataset are not equally represented. In these cases, bla **??** claims that classification accuracy is not a good metric to evaluate a model, 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 [20] is available in Sckit-learn, but there is a performance metric available there known as macro-average $F1$ that is suitable for evaluating imbalanced multiclass classification problems [21]. It averages the performance of classifying each individual class, in terms of precision and recall score. 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.

$$\text{Recall}(A) = \frac{\#\text{correct predictions}A}{\#\text{observations}A} \tag{2.4}$$

$$\text{Precision}(A) = \frac{\#\text{correct predictions}A}{\#\text{identified occurrences}A} \tag{2.5}$$

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

Another way to see if a high accuracy score is misleading is to analyze a so-called Confusion matrix. It shows the distribution of classifying observations for each class.

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. Regression algorithms can have either real-valued or discrete input variables. The model in eq. 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 the conversion proves useful, it is beyond the scope of this project.

Performance of regression models can be measured by computing the mean squared error (MSE) of the model 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, [22] writes that the primary goal of any regression model evaluation should be to minimize MSE.

2.2 Generalization

During the training process in supervised learning, a model is typically built based on its training data, and updated in order to reduce its training error. But 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 [23]. Both training error, how well a 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 generalizes to new data is overfitting and underfitting. These 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 [15].
- Underfitting: A model that performs poorly on both its training data and on generalization.

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 [24]. As a means to check whether or not a model suffers from overfitting, [25] suggests that when the test error of a model exceeds its training error, the model is overfitted to its training data. According to Davide [15] 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.

2.2.1 Cross-validation

An alternative, or perhaps complement, to the lock box approach as explained in 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. Although the name might be confusing, the validation is typically done by the test data and a validation dataset is not necessarily used. One cross-validation technique is called k-fold cross-validation. In k-fold cross-validation, the data is randomly 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 performance over 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 [15]. Furthermore, using $k = 10$ seems to be optimal when it comes to optimizing run-time for the test, limiting bias (underfitting) and variance (overfitting) [26]. 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 [15]. 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 [27].

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 [28]. It can also be applied to regression problems but the results from Breiman and Spector [29] indicate that there is little improvement from using this technique for regression problems.

2.2.2 Regularization

Another method used to overcome overfitting is regularization. This technique discourages complexity and flexibility of models by regularizing its coefficients toward zero. It can be used by The magnitude of the regularization can be controlled by a hyperparameter λ. Hyperparameters are model parameters whose value are set before the training process. The higher value of λ, the higher impact regularization has on the model, but high values on λ can result in underfitting and should therefore be controlled carefully [30]. 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 2.2.3).

$$Error_{L_1} = Error + \lambda \sum_{i=1}^{n} |\beta_i| \tag{2.8}$$

- 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 slowly reduced to zero.

$$Error_{L_2} = Error + \lambda \sum_{i=1}^{n} \beta_i^2 \tag{2.9}$$

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

$$Error_{L_1L_2} = Error + \lambda((1-\alpha) \sum_{i=1}^{n} |\beta_i| + \alpha \sum_{i=1}^{n} \beta_i^2) \tag{2.10}$$

A comparison of these techniques was made by [31] in their performance of modelling insulin sensitivity. The results demonstrate a slight advantage using Lasso and Elastic-net in terms of performance.

2.2.3 Feature selection

Feature selection is a process where irrelevant and redundant features are removed. The curse of dimensionality, which is brought up in 1.1, can be avoided by reducing the amount of features. Reducing the amount of features can lead to better generalization and performance [32], [33].

2.3 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 an ideal case where a single machine learning is best at solving all possible machine learning problems [29]. It is therefore interesting to study the effect 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 are named the same way as in [12]. Each family listed in 1.3 are covered.

2.3.1 Decision tree based learning

Decision tree algorithms uses a decision tree datastructure to solve classification and regression problems. Furthermore, 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 [34]. 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 etc. However, constructing such optimal decision trees has been proved to be a NP-complete problem [35].

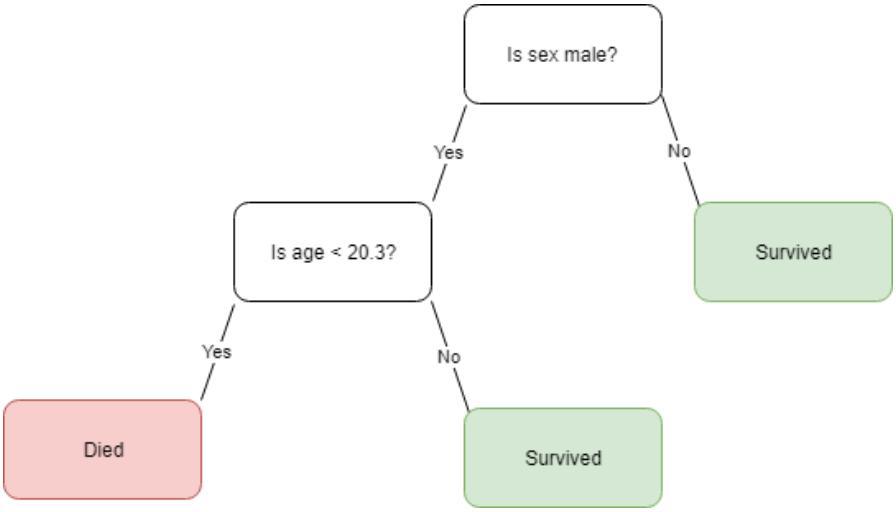

Figure 2.1: Example of a decision tree which predicts if a passenger survives a car crash or not.

Decision tree algorithms are allowed to capture nonlinear patterns in the training data, which makes them flexible, but also prone to overfitting [36]. Kotiantis [34] claims 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 [37]. 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 [37]. CART can solve both regression and classification problems and it is available in Scikit-learn [38].

2.3.2 Instance based learning

Instance-based learning (IBL) is about storing the provided training data, and using it to predict/classify a new observation. In other words, when a new observation is received, an IBL algorithm retrieves related observations from memory and uses it to predict/classify the 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 [39].

An example of an IBL algorithm is k-nearest neighbor (kNN). Generally, observations can be considered to be points in an n-dimensional space and kNN is based on the principle that these observations in a dataset are generally close to other observations

with similar properties. For a new observation, the algorithm finds the k closest neighbors and uses the properties of the k neighbors to classify the observation. kNN can be used to solve both regression and multiclass classification problems in Scikit-learn [40], [41].

Wu et. al [42] argue that kNN is suitable for multiclass classification problems. Another study by L. Zhong et. al [43] showed success in predicting CT images from MRI data by using kNN regression. However, [42] 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 [44] showed that an optimal choice of k grew linearly with an increase in the amount of training instances. 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 [34].

Wu et. al [42] talks about the issues of choosing a distance metric for kNN. They say that among several choices, it is desirable to have a distance measure in which a smaller value between two nodes implies a stronger connection. Scaling is another issue they talk about when it comes to distance metrics. 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.

2.3.3 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.

$$p(H|D) = \frac{p(H)p(D|H)}{p(D)} \tag{2.11}$$

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

Naïve Bayes classifiers assume that the features in a dataset are independent from one another. Which means that each feature contribute independently to classify an observation, regardless of any correlations that might exist among the features, which is why it is called naïve. Despite this naïve assumption and its ease of use, Naïve Bayes classifiers have proved successful, even when strong dependencies among features are present [45], [46]. 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 [47]. Naïve Bayes can be applied to multiclass classification problems [48].

The Gaussian Naiïve Bayes classifier assumes that the features have a gaussian (normal) data distribution.

2.3.4 Regression based learning

As stated in 1.3, only linear algorithms of Regression based learning agorithms are covered. A linear algorithm is one where its model is specified as a linear combination

of input features.

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 analysis refers to regression analysis: a number of statistical methods for estimating relationships among variables. This is a well-known tool in statistics and it is also used in 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 2.2 is an example of a trained regression based learning algorithm called Ordinary Least Squares (OLS). OLS is used in regression predictive modelling and a different regression based learning algorithm called Logistic regression (LR) can be used for classification problems.

Some of the assumptions listed by [12] that most regression based learning algorithms make of the training data:

- Linear behavior: If a target feature and an input feature have a linear relationship, it would look like a straight line when plotted against oneanother. Any non-linear dependencies between the target feature and other features are assumed to not be considered input features.
- Normal distribution: Assumes the data of the target feature is normally distributed.
- Outliers: Are expected to be handled.
- Data accuracy: Values of the target feature that are consider for classification/prediction are assumed to be deleted. For example, error states that are not to be classified.

In addition to applying cross-validation, overfitting in regression based learning algorithms can be thwarted by applying the regularization techniques as seen in 2.2.2. Feng et. al [49] chose λ, which is one of the regularization hyperparameters, based on which value of λ maximised their cross-validation scores. They did so by performing a grid-search (see 2.4) on λ.

2.3.5 Deep learning

Deep learning algorithms are those that aim to learn an artificial neural network (ANN) [12]. ANNs 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 hidden layers are there to transform the input to an output. 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 distinguishes edges in the picture, the second layer recognizes when these edges form complex shapes, and so on. Figure 2.2 depicts an example of a deep learning algorithm model with the same kind of problem as seen in 2.1.

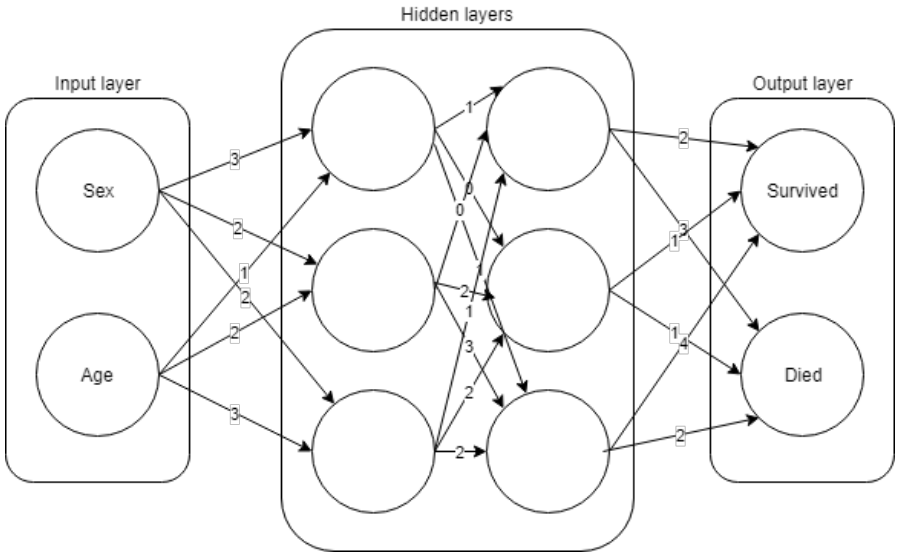

Figure 2.2: Example of a neural network with two hidden layers. The goal is to predict if a passenger survives a car crash or not.

The number of hidden layers h and the number of nodes per hidden layer N_h can be configured as hyperparameters in the different deep learning methods. So the question is, what is the optimal value of h and N_h? Heaton [50] writes that for most practical problems, having $h = 1$ is enough for most practical problems and that there is no theoretical 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 and the opposite may introduce overfitting and a significant increase in training time. The author 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 structure, but how are decisions made? Choices are made in the ANN by having each non-input node n_i process information from the previous node whose edge to n_i have the highest value among all nodes connected to n_i. In other words, which of the connected nodes to n_i to choose depends on the weights of the edges that connects them. How the weights are set correlates to the learning process, which is what distinguishes the different types of deep learning algorithms.

Backpropagation is a deep learning algorithm that can be used to solve supervised

learning problems. Broadly speaking, backpropagation updates the weights of the edges in the ANN based on a loss function. Whenever a "bad" classification/prediction is made (based on the loss function), the information is propagated backwards in the network from that classification/prediction and weights are adjusted accordingly.

2.3.6 Ensemble learning

Ensemble learning is based on the principle that combining the results of several classifiers/predictors into a collective score can produce better results than individual scores. Two independent studies indicate that Random forest, an ensemble learning algorithm, show high success when compared to non-ensemble algorithms[51], [52]. Random forest averages the results from several decision tree classifiers/predictors to achieve a result. Random forest can be applied to classification and regression problems [40].

2.4 Optimizing hyperparameters

Generally speaking, finding optimal hyperparameter values for various algorithms can be regarded as an experimental process. One brute-force approach, which according to [53] is the best way to verify optimality of hyperparameters, is to use a technique known as grid search. The technique systematically tests 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

2.5 Data preparation

2.5.1 Imbalanced data

A number of oversampling techniques that can be used to solve multiclass classification problems are suggested by **??**. However the suggested techniques are complex and can 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 Randomoversampler. The techniques can be used to solve multiclass classification problems in Imbalanced-learn, despite not being suggested by **??** as proper techniques to solve such problems.

Smote and Adasyn creates new synthetic points in the dataset while Randomoversampler creates duplicates. Synthetic points are points that are created from existing points but with small random 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 Randomoversampler [56].

Chapter 3

Method

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

3.1 Choice of machine learning software

Being new to the world of machine learning, the author suspected early on in the project that there would be a lot of new concepts to learn, which is why it was desirable to choose a machine learning software that is beginner-friendly. 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 it is claimed to be easy to use [57]. The plan was to stick with Scikit-learn, so long as it could be used to solve the aim of the thesis, and to change machine learning software when necessary.

3.2 Early-stage implementation and literature study

In order to learn about the relevant machine learning theory to solve the thesis subtasks, the author deemed it necessary to learn both from experimenting in Scikit-learn in parallell to conducting a literature study. The first implementation was one where CART was used to classify the iris dataset (see 2.1.1). Eventually, the actual dataset from 1.4 was used to see the effects of working with larger datasets and get an idea of how the different algorithms perform on the project subtasks. The dataset, being in six different Excel workbooks, was combined to one workbook. It was discovered that using the full dataset was computationally heavy. Since a lot of experimenting was done in the beginning to learn concepts, either the iris dataset or data from one of the weather stations was mainly used in the beginning. Selecting subsets of the data to work with is known as sampling in statistics. Experimenting with a sample of the full dataset and/or the Riris dataset was done throughout the project to perform learning experiments.

It was originally planned to conduct a literature study for a duration of about four weeks in the early stages of the project, but learning relevant machine learning theory

proved harder than estimated and as of such, the learning process continued throughout the project. A thesis objective could be established relatively early in the project, but finding the best means of achieving the objective proved hard. The following databases were used to find relevant books and academic papers to fulfill the aim of the study: Google Scholar, Google Books, Scopus and IEEE Xplore. At times, especially in the beginning of the project, it proved difficult for the author to machine learning concepts solely from academic sources. Therefore, non-peer-reviewed resources such as blogs and websites etc. were visited occasionally to attain a general understanding of key-concepts. If the author learned new concepts from non-peer-reviewed sources, its validity was checked in peer-reviewed material. A number of key-words were used to find relevant material in Google, Google Scholar, Google Books, Scopus and IEEE Xplore. Some of the keywords used:

- machine learning
- supervised learning
- multiclass classification
- regression
- cross-validation
- methodology
- overfitting
- performance

3.3 Data overview

The provided data described in 1.4 was studied from an early stage in the project. This was done primarily to determine if the subtask target features should be solved by regression or classification algorithms, but also to study the data distribution of each target feature to reveal how often errors occurr, 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 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 an algorithm trained on this dataset could perform well in predicting/classifying observations from the northern parts in sweden for example. However, the weather stations which provided the data for this project appear to be the only ones that have both the DSC111 and DST111 sensors. As of such, it was ultimately decided to not investigate if data from additional weather stations could 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 when needed.

3.4 Research approach

Among all of the algorithms in 2.3, finding the optimal ones that best solves the project subtasks may prove cumbersome. One approach could be to study the benefits and advantages of each algorithm or algorithm family in theory and predict which one is optimal for solving each of the project subtasks. In research approach theory, this is known as a deductive approach; where one makes a theoretical hypothesis and assumptions, and test and/or verify these empirically. Microsoft Azure and Scikit Learn provide intuitive guides of which specific algorithm to choose depending on size of dataset and which the type of problem [58], [59]. However similar algorithm-choosing guidelines seems to be few in academic resources. The author assumes that this is because performance are 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 optimal 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 familes, generally perform equally well. Thus if at least one algorithm from each algorithm family is represented, an optimal, or at least near-optimal, algorithm 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 [60]. 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 the given problem domain. Including one additional algorithm for comparison in Scikit-learn means one extra line of code, which makes spot-checking possible with Scikit-learn.

3.4.1 Choice of algorithms to evaluate

The algorithms to evaluate in spot-checking were chosen based on their interpretability, availability in Scikit-learn, ability to solve regression and classification problems, their general popularity and the 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 no
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
Backpropagation	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, Backpropagation and Random forest based on their ability to solve regression and classification problems, and based on a hypothesis made by the author from the information in 3.2, that they are among the most popular algorithms within their families. Three algorithms were chosen to represent regression based learning algorithms: OLS, Logistic regression and Lasso. 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. Lasso was chosen so that at least one regularization technique is evaulated. Lasso was chosen instead of Ridge or Elastic-net as regularization techniques since theory in 2.2.2 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.

Table 3.2 and 3.3 shows the algorithms used to solve regression and classification problems in this project. The algorithms are used with their default 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 also shown.

Name	Hyperparameter 1 default setting	Hyperparameter 2 default setting
CART	splitting critera: gini	
kNN	$k = 5$	distance metric: minkowski
OLS		
Lasso	$\lambda = 1$	
Backpropagation	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 also 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	
Backpropagation	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 rela-

tionship 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 3.2 where each algorithm runs with 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 keep deterministic behavior. The random state is obtained by using a seed, $seed = 7$ is an arbitrary choice.

- Holdout classification spot-checking: Same as the setup mentioned above but using the classification algorithms in 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. Although holdout was ultimately used as model validation technique in this project, the possibility of using k- fold instead was investigated using this setup. A value of $k = 10$ is used since it is supported in 2.2.1.

- Cross-validation classification spot-checking: Same as above apart from using stratified k-fold instead of regular k-fold and using the classification algorithms on default settings in **??**. Stratified k-fold is used to ensure that each class is equally represented in every fold.

- kNN optimization: The purpose of this experiment is to find optimal values for k in kNN. As brought up in 3.7, grid search is used as hyperparameter optimization technique, which applies to this experiment as well. Theory in 2.3.2 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 $k = 5$.

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

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

- Backpropagation optimization: This experiment deals with finding an optimal number of hidden nodes in Backpropagation. Similar to the choice of k in kNN, an optimal number of hidden nodes in Backpropagation seems to be somewhat of a mystery. A starting point, as mentioned in 2.3.5, 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, 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 optimal value for λ in Lasso. Apart from the standard value $\lambda = 1$ a range of values are tested.

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

3.6 Choice of model validation technique

From what the author learned from the literature review, it was planned to use a combination of holdout and k-fold as model validation technique: to perform an initial split of training/test data, use cross-validation to train a model and test it on the test data. It was believed that the combination of these techniques would yield high performance and generalization among the algorithms. However, it was discovered with using the built-in functionality in Scikit-learn to implement k-fold, it was not possible to train a model to be tested at a later point in Python code. This means it was possible to attain a cross-validation score from a given dataset, but to train a model using cross-validation and ultimately test it on a test dataset was not possible. The same reason is primarily why holdout was used as model validation technique instead of k-fold or stratified k-fold to achieve the final results. The way overfitting is detected in this project is to compare how a fitted model performs on its training dataset, in contrast 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. 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 k-fold was not used in achieving the final results, it was used in the early stages of the project and the data preparation process, since it was planned to be used throughout the project.

It was chosen to work with a value of $k = 10$ for both k-fold and stratified k-fold since this is supported in theory 2.2.1. When cross-validation was 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 the other target features. If holdout was used as model validation technique, a 80%%20 training/test split was used. This is an arbitrary choice, taken from the fact that an optimal split does not appear to exist (see 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 used in this project, it is 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 instead, which is heavier computation-wise, but this was mitigated by optimizing three hyperparameters in total: k in kNN, n.o. hidden neurons in Backpropagation and λ in Lasso (see 1.3).

3.8 Choice of performance measures

The author chose to use MSE to evaluate the performance of regression algorithms since this is supported in 2.1.2. Performance of classification algorithms were measured using macro $F1$ score (simply refererred to as $F1$ throughout the rest of this project). $F1$ was chosen due to the fact that it takes both precision and recall into account, and 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 still shown since it gives an intuititive idea of how an algorithm performs.

As described in 1.2, the project subtasks are about finding the algorithm that performs best in terms of both performance and generalization. How performance is measured has already been covered in this section, generalization has not. As mentioned in 2.2, overfitting is when a model fits its training data significantly better than its test data. Since theory in 2.2 suggests that underfitting is a result of a poor model fit, focus was set on overfitting in this project. It was decided to quantify overfitting rather than making it binary: yes overfitting or no overfitting. It was quantified by

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 [61] 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 method to perform data preparation. The author selected methods from different academic sources [61]–[63] that are thought to be relevant steps in the data preparation process for this project. The process is 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 1.3, this

project is about modelling non-error behavior which is why all of the reported errors are removed at this step. There may also be suspected errors such as outliers that are removed 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.

3. Data transformation: This step involves dealing with potential class imbalances and to see if input features can be transformed to improve overall performance.

3.9.2 Obtaining the results

After the data preparation process (see 3.9.1), 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 features f_i to f_t. This is done by calculating a correlation score of f_i to f_t. The correlation score is calculated with a so-called F-value which estimates the degree of linear dependency between two variables [64], [65].

2. Run i spot-checking experiments on the algorithms that can classify/predict f_t. 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.

   For regression tasks, holdout regression spot-checking experiments are used (see 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 the 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.

   For classification problems, holdout classification spot-checking experiments are used (see 3.5.1). Performance is measured in $F1$ macro average score on the test dataset $F1_{test}$. Overfitting $F1_{diff}$ 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: $F1_{diff} = F1_{test} - F1_{train}$. An overall score, which covers both performance and overfitting, is displayed as an accumulated score $P_{acc} = F1_{test} + F1_{diff}$. Accuracy is also displayed, but not evaluated on since it can be misleading in imbalanced datasets (see the classification section in literature study 2.1.1).

3. Once an optimal 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 1.3, only k in kNN, number of hidden nodes in Backpropagation and λ in Lasso are optimized. Lasso, Backpropagation and kNN optimization experiments are run as covered in 3.5.1.

4. Lastly, the optimal scores of each algorithm are listed and the algorithm, whose accumulated performance score is best, is identified as the top performing algorithm.

3.10 Tools

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

- Python: Programming language.
- Scikit-learn: Python machine learning library.
- Pandas: A Python library used in this project to read the dataset from Microsoft Excel workbooks.
- Pyplot: A Python library used to visualize 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 would probably take a long to perform manually.
- Sublime text: Source code editor.
- Git: Version control tool.

Chapter 4

Data preparation process

This chapter describes the process of preparing the data.

4.1 Data cleaning

In the provided guidelines (see 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 the 1429 that the average difference between the DST111 and Track Ice Road Surface 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 above 40°C. This indicates that something may have been wrong with DST111, or some other instrument at the time and thus, the 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 higher still 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 1.3, only non-error features are used in this project. As of such, every observation where at least one error is reported by any sensor, are removed. Figure 4.1 shows that the DSC111 alone was malfunctioning for unknown reasons in more than 50000 observations.

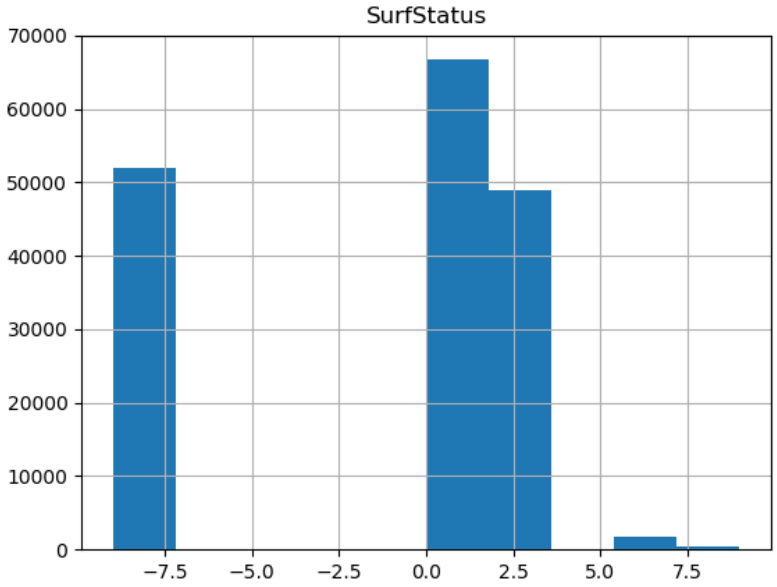

Figure 4.1: Histogram showing the distribution of the different surface status types (see 1.2 to see what each code means).

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 time should be excluded as a possible input feature, primarily because the values are higher than the rest of the features. Some supervised learning algorithms, such as kNN, are sensitive to scaling (see 2.3.2). Time 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: $\text{time} = 12312330, \text{SurfTemp} = 7.1, ...\text{Friction} = 0.74$.

Although the scaling of the time feature is significantly different from the other input features, it seems to be correlated with other features. Figure 4.2 shows how every feature is related to one another, it indicates that features such as time and friction are correlated with time. The correlation may come as no surprise since the northern hemisphere is colder during winter-time, which means cold temperatures and low friction, and the other way around during warmer periods.

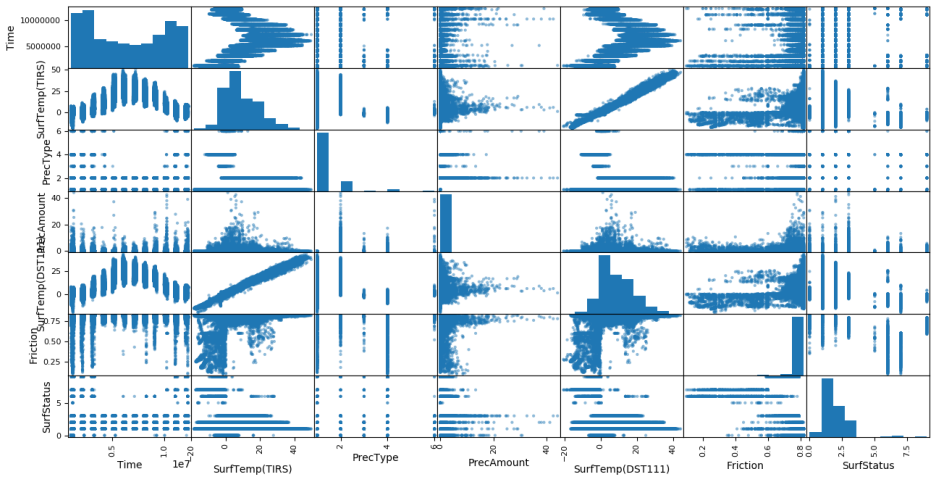

Figure 4.2: Depicts how the features are correlated to oneanother.

To test the relevance of using time as input feature, an experiment was carried out to predict Track Ice Road Sensor road surface temperature once with time as input feature, and once without. The experiment runs the Cross-validation regression spot-checking setup (see 3.5.1). Table 4.1 shows the result from the experiment.

Table 4.1: Experiment to see if time 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
Backpropagation	1.06	769096.72
Lasso	1.25	1.24
Random forest	1.13	1.26
Average total	1.19	128184.50

The results indicate that OLS, Lasso and Random forest achieve slightly better performance by using time as input feature, whereas CART, kNN, Random forest and especially Backpropagation, 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 this data was not ruled out yet. Section 4.3 deals with testing if time can be scaled down to similar levels of other features to see if it improves overall performance or not.

4.3 Data transformation

4.3.1 Rescaling time

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

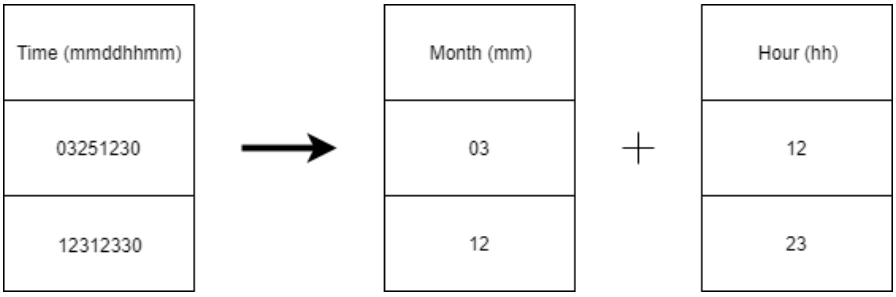

Figure 4.3: Shows how time as feature is transformed to two new features: month and hour.

The author assumes that month and time of day affect changes in weather conditions more than separate days within a month. Figure 4.5 shows the correlations among the features with month and hour used instead of time.

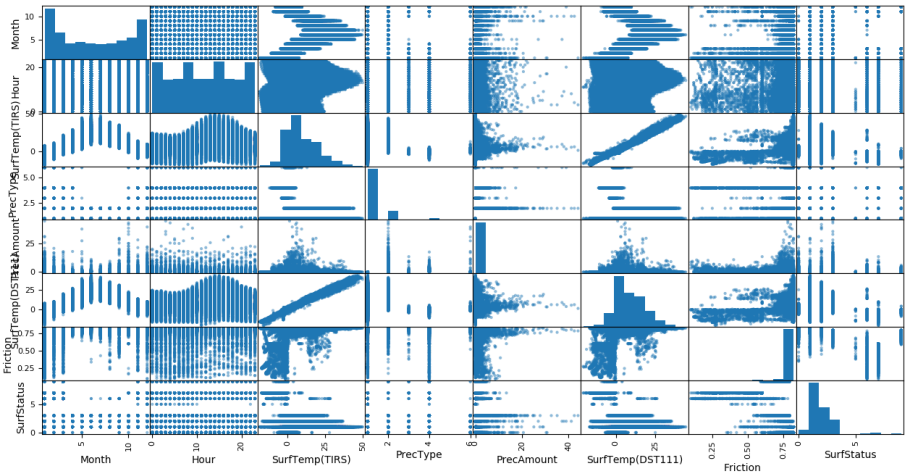

Figure 4.4: Correlations among the different features where month and hour are two new features that have replaced time.

An experiment with Cross-validation regression spot-checking setup was used to study the effects of this new transformation. Table 4.2 shows the result.

Table 4.2: Experiment to see the effects of transforming the time feature is relevant to use as input feature.

Algorithm	MSE using neither time nor new features	MSE using time	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
Backpropagation	1.06	769096.72	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	128184.50	1.27	1.19	1.20

Performance is improved for all algorithms when the new features are used as a way to represent time rather than using the old feature. However, not using time, hour or month, seem to have slightly higher performance than using both hour and month, but similar to using either of them. Although little to no overall performance improvement was made, it was decided to use the transformed input features in the default features for spot checking. The reason is that this experiment is set up to test Track Ice Road Surface road temperature alone. The new features may prove more relevant with the other target features and different hyperparameter settings etc.

4.3.2 Handling class imbalance

From what is seen in 4.1, road surface status seem to suffer from class imbalance. But classifying road surface status is not a goal in this project. It is interesting to see

if precipitation type suffers the same symptom since classifying that is a goal in this project. Figure 4.5 shows the distribution of precipitation type in the dataset.

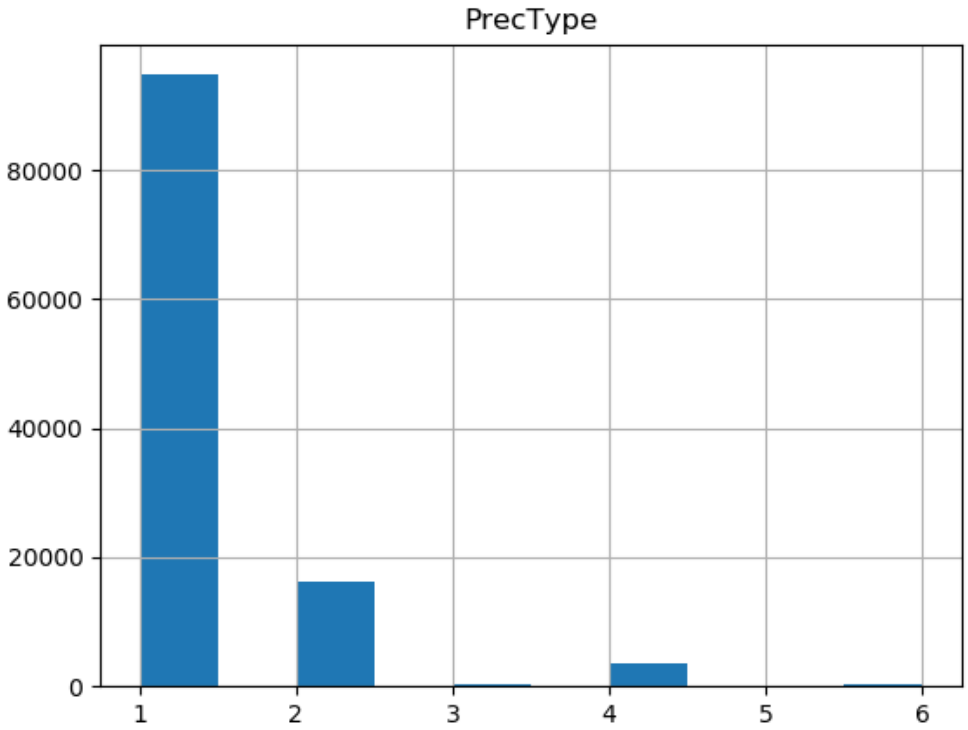

Figure 4.5: Histogram showing the distribution of the different types of precipitation in the data (see 4.3 to see what each code means).

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

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

Code	Precipitation type	no 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 4.3, precipitation also suffers a significant class imbalance.

The biggest difference is between no precipitation and rain and snow mixed, the former appear roughly 300 times more than that the latter.

An experiment was carried out to see if the class imbalance problem can be mitigated by using random oversampling as oversampling technique (see 2.5.1. Random oversampling and Smote was tested in this experiment. Smote runs on the default settings as seen in **??**. Both random oversampling and Smote are compared to a scenario where oversampling is not used. All three scenarios run the holdout classification spot-checking setup with feature engineered features. Holdout was used instead of k-fold in both scenarios since it proved cumbersome to integrate oversampling with k-fold, plus a confusion matrix can be obtained when the holdout method is used in Scikit-learn, which can make the results easier to interpret.

Table 4.4: Results from not using any oversampling techniques.

Algorithm	Accuracy	$\overline{Precision}$	$\overline{Recall}$	$\overline{F_1}$
LR	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
Backpropagation	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	$\overline{Precision}$	$\overline{Recall}$	$\overline{F_1}$
LR	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
Backpropagation	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	$\overline{Precision}$	$\overline{Recall}$	$\overline{F_1}$
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
Backpropagation	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 F_1 score is similar to the case when oversampling is not used. In the case of using Smote as oversampling technique as shown in 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: F_1, 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 Summary

A solution to each of the project subtasks are covered in the following sections. The results are achieved methodigally for each subtask (see experimental methodology 3.9.2). A summary of the methodology of finding an optimal 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 attain a ranking of the input features.
2. Run i spot-checking experiments to find optimal performances when the amount of top input features used vary. An accumulated score P_{acc} is calculated which takes both performance and overfitting into account.
3. Optimize Lasso, Backpropagation and kNN using their optimal choice of top input features.
4. Analyze the best scores for each algorithm using their optimal input features and settings. The algorithm whose accumulated performance score is highest is identified as the top performing algorithm.

Table 5.1 shows a summary of the top performing algorithms that were found for predicting/classifying 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	Optimal settings	Best performance
Optic eye	model precipitation type	classification problem	CART	Scikit default	$(Accuracy, F1_{test}) = (0.84, 0.46)$
Optic eye	model precipitation amount	regression problem	kNN	$K = 64$	$MSE_{test} = 0.54$
Track Ice Road Sensor	model TIRS road surface temperature	regression problem	Backpropagation	n.o. hidden nodes: 64	$MSE_{test} = 0.88$
DST111	model DST111 road surface temperature	regression problem	Random forest	Scikit default	$MSE_{test} = 10.16$

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

5.2 Predicting road surface temperature (Track Ice road sensor)

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 5.2 shows that DST111 most relevant 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)
Backpropagation	(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 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 number of input features is considered optimal.

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
Backpropagation	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 5.4 show that OLS and Backpropagation has best accumulated scores when using the top five features, whereas kNN is optimal when the top two features are used. CART, Lasso and Random forest have optimal scores in using the top one feature.

5.2.3 Optimizing hyperparameters

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

Algorithm	Default hyperparameter setting	Optimal setting	Default performance P_{acc}	Optimal performance (MSE_{test}, MSE_{diff})	Optimal performance P_{acc}
kNN	$k = 5$	$k = 64$	1.22	(1.00, 0.04)	1.04
Backpropagation	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 5.5, accumulated scores were improved for kNN and Lasso when optimal hyperparameter settings were used. Backpropagation 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 optimal choice and thus an optimal overall performance was obtained.

5.2.4 Results and analysis

Table 5.6: Shows the overall optimal settings and performances for each of the algorithms in predicting TIRS road surface temperature.

Algorithm	Optimal settings	Input features used	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
Backpropagation	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 Backpropagation 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. Backpropagation have a performance score of $MSE_{test} = 0.88$ which means that in predicting TIRS road surface temperature, Backpropagation was on average off by 0.88 from the actual values.

5.3 Classifying precipitation type (Optic Eye)

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 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.

5.3.1 Spot-checking

Table 5.8: Results from spot-checking experiment on the top features for classifying precipitation type. The results are shown as a triple: (Accuracy, $F1_{test}$, $F1_{diff}$) where $F1_{test}$ represents performance and $F1_{diff} = F1_{test} - F1_{train}$ shows the degree of overfitting, larger values of $F1_{diff}$ 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)
Backpropagation	(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 $P_{acc} = F1_{test} - F1_{diff}$. Top results for each algorithm are highlighted. In case of ties, the one using the fewest number of input features is considered optimal.

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
Backpropagation	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 5.9 show that kNN, CART, Naïve bayes and Random forest perform best when using the top five features, whereas Backpropagation and Logistic regression have optimal performances when using the top four and top one features respectively.

5.3.2 Optimizing hyperparameters

Table 5.10: Shows the effect of optimizing the hyperparameters of kNN, Backpropagation and Lasso using their top input features

Algorithm	Default hyperparameter setting	Optimal setting	Default performance P_{acc}	Optimal performance (accuracy, $F1_{test}$, $F1_{diff}$)	Optimal performance P_{acc}
kNN	$k = 5$	$k = 1$	0.45	(0.81, 0.40, -0.47)	0.87
Backpropagation	n.o. hidden nodes: 100	n.o. hidden nodes: 100	0.35	(0.85, 0.33, -0.01)	0.34

As shown in 5.10, the optimal settings for Backpropagation is the same as its default settings. Its overall performance was slightly reduced in comparison to the results in 5.8 and 5.9. This is believed to be due to non-deterministic behavior of Backpropagation. 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 5.9.

5.3.3 Results and analysis

Table 5.11: Shows the overall optimal settings and performances for each of the algorithms in classifiying precipitation type.

Algorithm	Optimal settings	Input features used	Best performance (Accuracy, $F1_{test}$, $F1_{diff}$)	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
Backpropagation	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 5.6 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 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 $F1_{test} = 0.46$ and $accuracy = 0.82$. This means that out of all observations in the test dataset, CART correctly classified 82% of the observations correctly. However, $F1 = 0.46$ indicates that the accuracy score is misleading. The recall scores in 5.12 reveal that CART performed well in predicting no precipitation: 90%, and that lower scores were obtained with classifying the other precipitation types. The lowest recall score is predicting rain and snow mixed which were 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 optimal setup as shown in 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)
Backpropagation	(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 optimal.

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	**0.89**
kNN	0.75	0.68	0.73	0.77	**0.68**
Backpropagation	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 5.15 indicate that all algorithms performs at best when the top one feature is used.

5.4.3 Optimizing hyperparameters

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

Algorithm	Default hyperparameter setting	Optimal setting	Default performance P_{acc}	Optimal performance (MSE_{test},MSE_{diff})	Optimal performance P_{acc}
kNN	$k = 5$	$k = 64$	0.68	(0.54, -0.05)	0.49
Backpropagation	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 optimal hyperparameter settings while Backpropagation produced the same result.

5.4.4 Results and analysis

Table 5.17: Shows the overall optimal settings and performances for each of the algorithms in predicting precipitation amount.

Algorithm	Optimal settings	Input features used	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
Backpropagation	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 5.17, Backpropagation 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 backpropagation 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 0.54 off from the actual values.

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 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 $MSE_{diff} = MSE_{test} - MSE_{train}$ 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)
Backpropagation	(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 features is considered optimal.

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
Backpropagation	**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 5.20 show that OLS, kNN, Backpropagation and Lasso performs at best when all top six features are used, while CART and Random forest benefits most

from using the top five features.

5.5.3 Optimizing hyperparameters

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

Algorithm	Default hyperparameter setting	Optimal setting	Default performance P_{acc}	Optimal performance (MSE_{test},MSE_{diff})	Optimal performance P_{acc}
kNN	$k = 5$	$k = 32$	12.48	(10.69, 0.43)	11.12
Backpropagation	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 optimal hyperparameter settings.

5.5.4 Results and analysis

Table 5.22: Shows the overall optimal settings and performances for each of the algorithms in predicting DST111 road surface temperature.

Algorithm	Optimal settings	Input features used	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
Backpropagation	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 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 10.16 from the actual values.

Chapter 6

Summary

This chapter aims at summarizing the objective of this project as well as the results obtained for the given objective.

6.1 Recap and solutions to project subtasks

This project has dealt with investigating whether or not road weather station 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.
- Track Ice Road Sensor: Dug into the road. Measures road surface temperature.
- DST111: Measures road surface temperature via infrared.

The idea was to try to model each sensor from data provided by the others, except for Track Ice Road Sensor since this may be removed in the future. In addition to the sensor above, any model may be trained using data from the following sensors 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 1.2 is as follows:
"The objective is to find optimal supervised learning models, in terms of performance and generalization, which models the behavior of the following sensors: Optic Eye, Track Ice Road Sensor and DST111 where each sensor is trained from observations made from other the other sensors but its own. In addition to the forementioned sensors, except for Track Ice Road Sensor, any model may train on the following input features as well: measurement timestamp, road friction and road surface condition."
The objective was broken down into four subtasks. Each subtask and their solutions are covered in the following subsections.

6.1.1 Classifying precipitation type

The first subtask mentioned in 1.2 is:

Find the algorithm among supervised learning algorithms, that can be used to classify **precipitation type**, whose performance and generalization score is best. The data may contain the following input features:

- measurement timestamp
- DST111 road surface temperature
- road friction
- road surface condition

6.1.2 Predicting precipitation amount

The second subtask mentioned in 1.2 is:

Find the algorithm among supervised learning algorithms, that can be used to predict **precipitation amount**, whose performance and generalization score is best. The data may contain the following input features:

- measurement timestamp
- DST111 road surface temperature
- road friction
- road surface condition

The results from 5.4.4 shows that the best algorithm in terms of performance and generalization in predicting precipitation amount is kNN which obtained a performance score of $MSE_{test} = 0.54$.

6.1.3 Predicting Track Ice Road Sensor road surface temperature

The third subtask mentioned in 1.2 is:

Find the algorithm among supervised learning algorithms, that can be used to predict **Track Ice Road Sensor road surface temperature**, whose performance and generalization score is best. The data may contain the following input features:

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

- road surface condition

It was found in 5.2.4 that the best algorithm in terms of performance and generalization for predicting TIRS road surface temperature is Backpropagation which obtained a performance score of $MSE_{test} = 0.88$.

6.1.4 Predicting DST111 road surface temperature

The fourth subtask mentioned in 1.2 is:

Find the algorithm among supervised learning algorithms, that can be used to predict **DST111 road surface temperature**, whose performance and generalization score is best. The data may contain the following input features:

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

From the results in 5.5.4 it was found that Random forest was the top performing algorithm in terms of performance and generalization. It obtained a performance score of $MSE_{test} = 10.16$.

Chapter 7

Conclusions and discussion

This chapter presents conclusions connected to the aim of this project, as well as an informal discussion on the results obtained.

7.1 Conclusions

This aim of this project was to find optimal supervised learning models to model the behavior of three sensors: Optic Eye, Track Ice Road Sensor 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 types of measurements in total: precipitation type, precipitation amount, TIRS road surface temperature and DST111 temperature. The objective was broken down into four subtasks which aimed at finding optimal algorithms to model each of the measurements of 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 CART for precipitation type and kNN for precipitation amount. An accuracy score of 0.84 and macro $F1$ score of 0.46 were obtained in modelling precipitation type using Scikit-learn default settings. Precipitation amount was best modelled using kNN, obtaining a performance score $MSE = 0.54$ setting $k = 64$ in Scikit-learn.

The two remaining sensors measuring road surface temperature: Track Ice Road Sensor and DST111, were best modelled using Backpropagation and Random forest respectively. Backpropagation was set to use 64 hidden nodes with which a performance score $MSE = 0.88$ was obtained in modelling TIRS road surface temperature. As for modelling DST111 road surface temperature, the best model was obtained by using Random forest on Scikit-learn default settings with a performance score $MSE = 10.16$.

7.2 Discussion

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, DST111 road surface temperature shows a higher error rate than precipitation amount and Track Ice Road Sensor road surface temperature. Since DST111 and TIRS both measure road surface temperature, it is assumed that the model for DST111 generally performs poorly and thus that the behavior of DST111 is hard to model with the given input features. Since any algorithm which models TIRS in this project can use the measurements from DST111, which proves to have strong linear correlation to TIRS, the author assumes that if TIRS road surface temperature could be used as input feature to model DST111, a better performance could be obtained.

As for the classification task of classifying precipitation type, the author deems that its performance should be evaluated on its $F1$ score rather than on its accuracy score in this project. The results from 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 $F1$ 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 solved in this project.

In hindsight, the author thinks that the accumulated performance score presented in 3.9.2 can be improved by not allowing negative values in the overfitting score. An overfitting score of 0 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 suspects that a negative overfitting score is not necessarily better than one around zero, but it can have a significant positive impact on the accumulated performance score. This may result in non-optimal algorithms having higher accumulated performance scores than optimal ones, and thus being identified as top performers.

7.3 Recommendations

The author recommends Trafikverket to refrain from modelling DST111 road surface temperature using the given algorithm and algorithm settings, but to investigate the possibility of modelling Track Ice Road Sensor road surface temperature and precipitation amount using the algorithms and algorithm settings seen in **??**. 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, either by collecting more data from the less represented classes, or to see if the problem can be solved using similar techniques as the ones used in this project (see attempts in handling class imbalance in 4.3.2).

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] ——, (2017). Vinterväghållning, [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] ——, *Nästa generation VägVäderinformationsSystem (VViS)*.

[8] Vaisala. (). 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).

[9] Trafikverket, *Road weather information systems*.

[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] scikit-learn developers. (). Supervised Learning, [Online]. Available: http://scikit-learn.org/stable/supervised_learning.html (visited on 04/05/2018).

[14] M. Skocik, J. Collins, C. Callahan-Flintoft, H. Bowman, and B. Wyble, "I TRIED A BUNCH OF THINGS: THE DANGERS OF UNEXPECTED OVERFITTING IN CLASSIFICATION", *bioRxiv*, 2016. DOI: 10.1101/078816. eprint: https://www.biorxiv.org/content/early/2016/10/03/078816.full.pdf. [Online]. Available: https://www.biorxiv.org/content/early/2016/10/03/078816.

[15] 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.

[16] 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.

[17] 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).

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

[19] R. Fisher. (). Iris Data Set, [Online]. Available: https://archive.ics.uci.edu/ml/datasets/iris (visited on 03/19/2018).

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

[21] Scikit-learn. (). 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).

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] "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.

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

[29] "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.

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

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

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

[39] 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.

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

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] H. Zhang, "The Optimality of Naïve Bayes", in *In FLAIRS2004 conference*, 2004.

[47] 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.

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

[49] Y. Feng, J. Fan, Y. Feng, and Y. Wu, "NETWORK EXPLORATION VIA THE ADAPTIVE LASSO AND SCAD PENALTIES.", *ANNALS OF APPLIED STATISTICS*, vol. 3, no. 2, pp. 521 –541, n.d. ISSN: 19326157. [Online]. Available: http://proxy.lib.ltu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edswsc&AN=000271979600002&lang=sv&site=eds-live&scope=site.

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

[51] 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.

[52] 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.

[53] 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.

[54] imbalanced learn. (). 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. (). 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. (). 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] Microsoft. (). 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).

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

[60] J. Brownlee. (). 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).

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

[62] 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.

[63] 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.

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

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