Towards impressive titles

Tobias Axelsson

30 april 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 supervised learning algorithms that best models the observations made by the following sensors: Optic Eye, Track Ice Road Sensor, DST111 and DSC111 in terms of performance with respect to generalization. The objective can be broken down into four tasks:

1. Find the highest possible accuracy score among supervised learning algorithms that can be used to model **precipitation type** with respect to generalization. The data may contain the following input features:
    - measurement timestamp
    - DST111 road surface temperature
    - road friction
    - road surface condition

2. Find the lowest possible mean squared error score among supervised learning algorithms that can be used to model **precipitation amount** with respect to generalization. The training data may contain the following input features:

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

3. Find the lowest possible mean squared error score among supervised learning algorithms that can be used to model **Track Ice Road Sensor road surface temperature** with respect to generalization. The training 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 lowest possible mean squared error score among supervised learning algorithms that can be used to model **DST111 road surface temperature** with respect to generalization. The training data may contain the following input features:

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

The algorithm models are built from 171425 observations provided by six weather stations along state road e6 in Sweden, measuring every 30 minutes over two years.

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

At least one algorithm per algorithm family, are considered in this project.

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.

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 F_1 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 F_1 scores are calculated for a multiclass classification problem on one if its classes A.

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

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

$$F_1(A) = 2 \cdot \frac{\text{Recall}(A) \cdot \text{Precision}(A)}{\text{Recall}(A) + \text{Precision}(A)} \tag{2.6}$$

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. According to [32], a reduction in dimensionality can also lead to better generalization.

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 [33]. 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 [34].

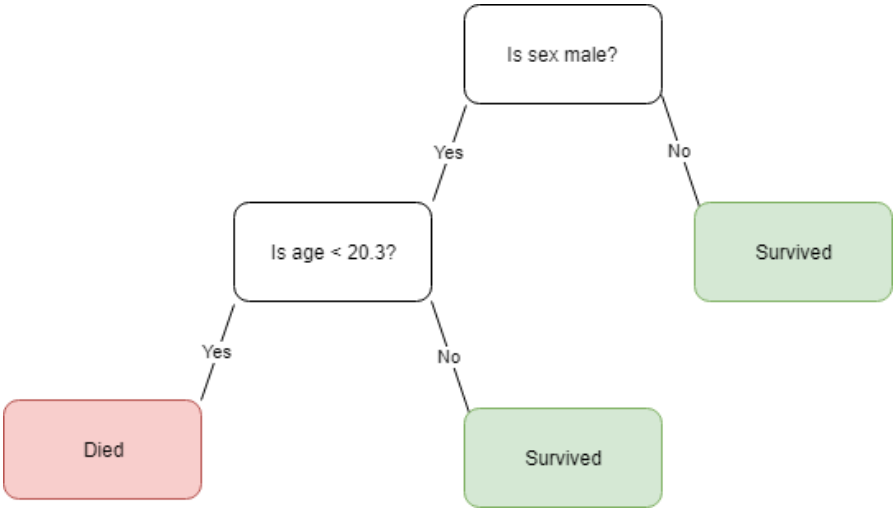

Figure 2.1: Example of a decision tree which predicts if a passenger survives a car crash or not.

Decision tree algorithms are allowed to capture nonlinear patterns in the training data, which makes them flexible, but also prone to overfitting [35]. Kotiantis [33] 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 [36]. 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 [36]. CART can solve both regression and classification problems and it is available in Scikit-learn [37].

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 [38].

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 [39], [40].

Wu et. al [41] argue that kNN is suitable for multiclass classification problems. Another study by L. Zhong et. al [42] showed success in predicting CT images from MRI data by using kNN regression. However, [41] 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 [43] 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 amount of training data. However, scientific methods for choosing a specific optimal value of k are lacking [33].

Wu et. al [41] 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 [44], [45]. 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 [46]. Naïve Bayes can be applied to multiclass classification problems [47].

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 [48] 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 2.5) 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.

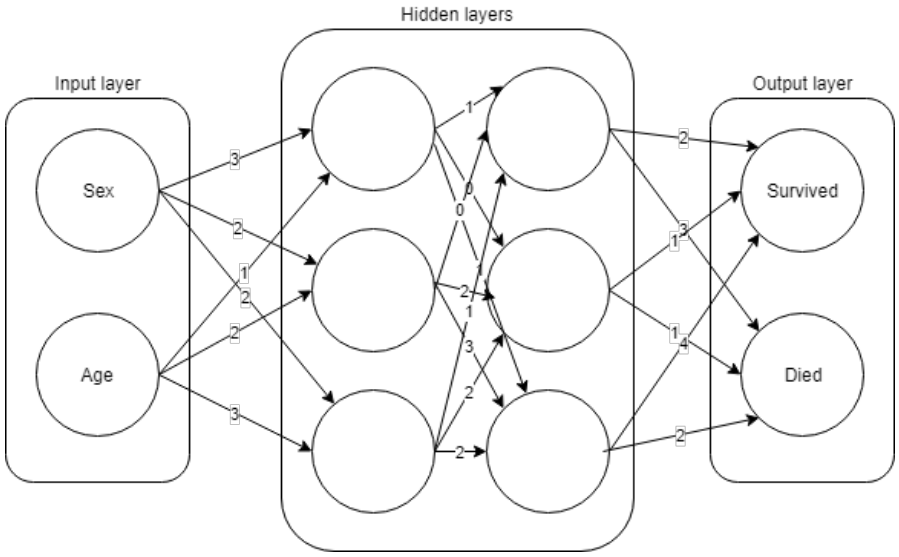

Figure 2.2: Example of a neural network with two hidden layers. The goal is to predict if a passenger survives a car crash or not.

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 [49] 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[50], [51]. 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 [39].

2.4 Algorithm selection

Selecting the optimal algorithms among all of the algorithms in 2.3 may prove cumbersome. One approach could be to study the benefits and advantages of each algorithm or algorithm family in detail and decide beforehand which one to choose. 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 [52], [53]. However similar algorithm-choosing guides seems to be few in academic papers, perhaps because there are many different situations, hyperparameters, data sizes etc .

Another approach to selecting an optimal algorithm is experimental.

2.5 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 [54] 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

2.6 Data preparation

2.6.1 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 [55]. 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 [56]. 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 [57].

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, which are ordered chronologically.

3.1 Data overview and test-implementation

3.2 Literature study

3.3 Choice of algorithms

3.4 Choice of model validation technique

3.5 Data preparation experiment setup

3.6 Experimental methodology

3.6.1 Data preparation

- Remove all errors
- Study data and take actions (class imbalance), remove suspicious outliers

Chapter 4

Data preparation

This chapter describes the process of preparing the data.

4.1 Data analysis

4.2 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.

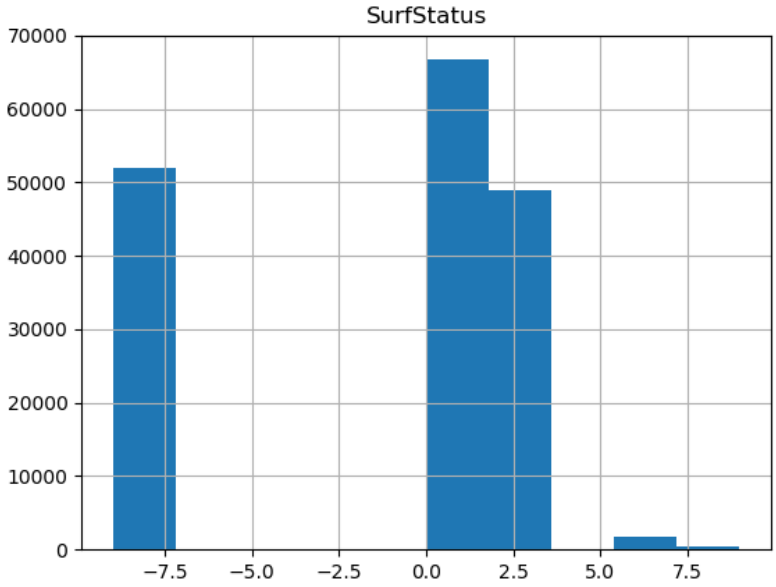

Figure 4.1: Histogram showing the distribution of the different surface status types (see 1.2 to see what each code means).

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. The observations were removed programmatically by implementation of specific measures in Visual Basic.

4.3 Feature selection

It was decided to investigate if time should be excluded as a possible input feature, primarily because the values it presents 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: 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.

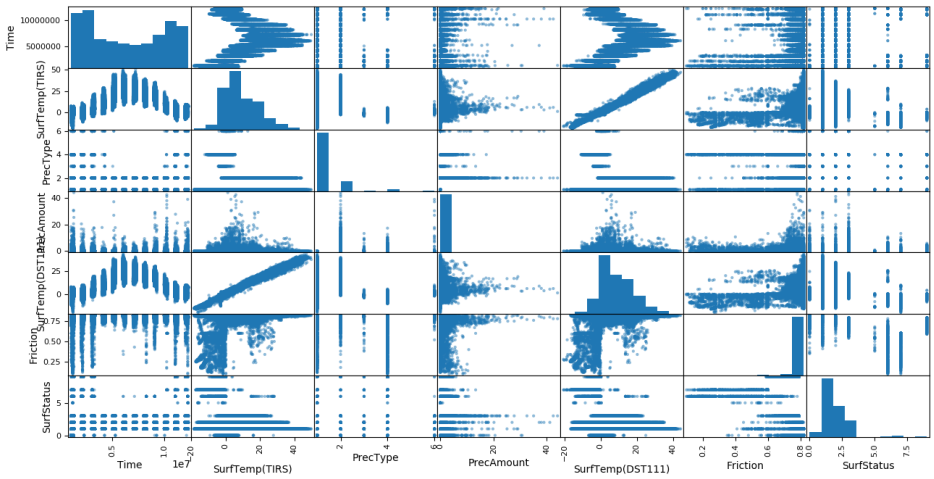

Figure 4.2: Depicts how the features are correlated to oneanother.

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. All supervised algorithms run on default configurations, a training/test split of 80%/20% was used and the dataset is shuffled randomly, but same in both runs by using a fixed seed value. 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 is not ruled out yet. Section 4.4 deals with testing if time can be scaled down to similar levels of other features to see if it improves overall performance or not.

4.4 Data transformation

4.4.1 Rescaling time

Table 4.1 shows significant improvement in overall performance in not using time as input feature. However, figure 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 would 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.

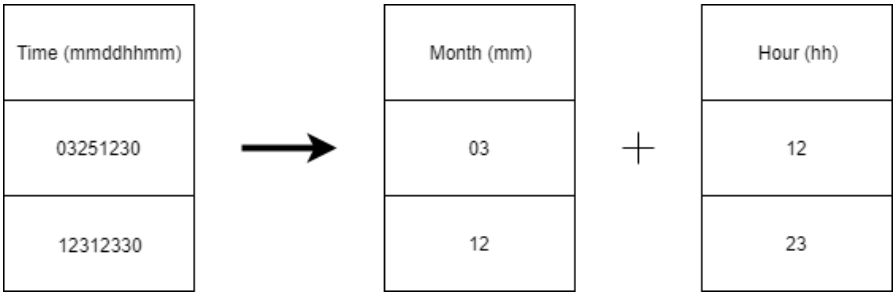

Figure 4.3: Shows how time as feature is transformed to two new features: month and hour.

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.

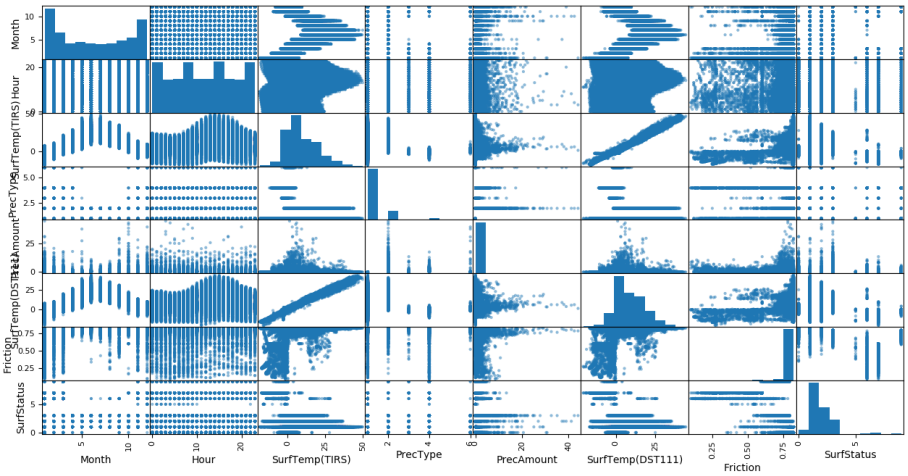

Figure 4.4: Correlations among the different features where month and hour are two new features that have replaced time.

All that's left is to see the effects of the transformation. An experiment with default setting was set up 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. The data transformation was done by implementation of specific measures in Visual Basic.

4.4.2 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.

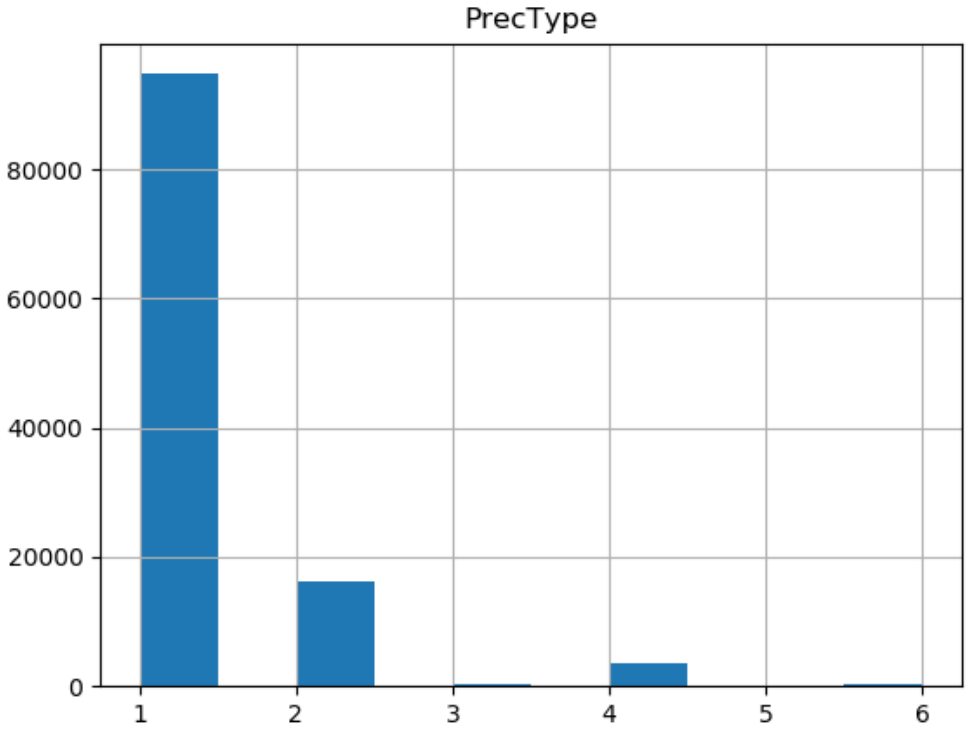

Figure 4.5: Histogram showing the distribution of the different types of precipitation in the data (see 4.3 to see what each code means).

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 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 **??**. 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 experiment setup with feature engineered features and all algorithms on default settings 3.5. 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 algorithm performance, and since no improvement was made on the collective score of precision and recall: F_1, and overall accuracy was reduced when either 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 tasks 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

Describe the process of collecting data, training and implementing machine learning algorithms with different methods.

5.1 Predicting road surface temperature (Track Ice road sensor)

5.1.1 Input features correlation rankings

Table 5.1: Relevancy of each possible input feature to the target feature: TIRS road surface temperature.

Relevancy ranking	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

5.1.2 Spot checking

Table 5.2: Results from spot-checking test on the top 7 features for predicting TIRS road surface temperature.

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**	**1.19**	**1.19**	1.20	1.22	1.22	1.22
CART	1.36	1.35	1.31	1.05	1.03	1.02	**1.01**
kNN	0.94	0.93	**0.92**	1.08	1.16	1.16	1.21
Backpropagation	0.90	**0.86**	**0.86**	0.97	1.00	1.02	1.01
Lasso	**1.24**	**1.24**	**1.24**	**1.24**	**1.24**	**1.24**	**1.24**
Random forest	1.01	**1.00**	1.01	1.01	1.01	1.01	1.01

Table 5.3: Shows the difference between test error and training error ($MSE_{test} - MSE_{train}$) over the top features. The difference is denoted as OF and top results for each algorithm are highlighted. Larger values indicate overfitting

Algorithm	OF top 7	OF top 6	OF top 5	OF top 4	OF top 3	OF top 2	OF top 1
OLS	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**
CART	1.10	1.08	1.03	0.30	0.15	0.09	**0.05**
kNN	0.32	0.32	0.31	0.18	0.09	0.06	**0.04**
Backpropagation	**0.01**	0.02	**0.01**	0.03	0.02	0.02	**0.01**
Lasso	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**
Random forest	0.65	0.66	0.64	0.23	0.12	0.08	**0.05**

5.1.3 Mid-analysis

5.1.4 Improving results

5.1.5 Analysis

5.2 Classifying precipitation type (Optic Eye)

5.2.1 Pre-analysis

5.2.2 Spot checking

5.2.3 Mid-analysis

5.2.4 Improving results

5.2.5 Analysis

5.3 Predicting precipitation amount (Optic Eye)

5.3.1 Input features correlation rankings

Table 5.4: Relevancy of each possible input feature to the target feature: precipitation amount.

Relevancy ranking	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

5.3.2 Spot checking

Table 5.5: Results from spot-checking test on the top features for predicting precipitation amount.

Algorithm	MSE top 5	MSE top 4	MSE top 3	MSE top 2	MSE top 1
OLS	**0.58**	**0.58**	**0.58**	**0.58**	**0.58**
CART	1.01	0.63	0.98	0.98	**0.62**
kNN	0.60	0.61	**0.58**	0.62	0.63
Backpropagation	0.56	0.57	**0.55**	0.56	**0.55**
Lasso	**0.61**	**0.61**	**0.61**	**0.61**	**0.61**
Random forest	0.67	0.59	0.68	0.71	**0.57**

Table 5.6: Shows the difference between test error and training error ($MSE_{test} - MSE_{train}$) over the top features. The difference is denoted as OF and top results for each algorithm are highlighted. Larger values indicate overfitting

Algorithm	OF top 5	OF top 4	OF top 3	OF top 2	OF top 1
OLS	**-0.06**	**-0.06**	**-0.06**	**-0.06**	**-0.06**
CART	0.94	0.18	0.91	0.80	**0.17**
kNN	0.15	0.07	0.15	0.15	**0.05**
Backpropagation	-0.06	**-0.05**	**-0.05**	-0.06	-0.06
Lasso	**-0.05**	**-0.05**	**-0.05**	**-0.05**	**-0.05**
Random forest	0.52	0.13	0.51	0.46	**0.11**

5.3.3 Mid-analysis

5.3.4 Improving results

5.3.5 Analysis

5.4 Predicting road surface temperature (DST111)

5.4.1 Spot checking

5.4.2 Mid-analysis

5.4.3 Improving results

5.4.4 Analysis

Chapter 6

Conclusions and recommendations

6.1 Conclusions

6.2 Recommendations

Chapter 7

Discussion

7.1 Thesis process

7.2 Validity and reliability

Validity and reliability of the conclusions. Needed?

7.3 Future work

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] 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.

[34] 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.

[35] 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.

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

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

[38] 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.

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

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.

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

[48] 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.

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

[50] 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.

[51] 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.

[52] 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).

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

[54] 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.

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

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

[57] 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).