Evaluating Incremental Machine Learning Models for Road Condition Classification

Degree Project Report in Computer Science and Engineering - LMTX38

David Svantesson & Julia Hansen

DEPARTMENT OF Computer Science and Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024
www.chalmers.se | www.gu.se

Degree project report in Computer science and engineering
- LMTX38 2024

Evaluating Incremental Machine Learning Models for Road Condition Classification

David Svantesson
Julia Hansen

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Degree Project Report in Computer Science and Engineering - LMTX38

Evaluating Incremental Machine Learning Models for Road Condition Classification
David Svantesson & Julia Hansen

Supervisor: Adam Breitholtz, Chalmers Data Science och AI
Supervisor: Pontus Andersson, Klimator AB
Examiner: Jonas, Almström Duregård

Degree project report 2024
Department of Computer Science and Engineering
Chalmers University of Technology | University of Gothenburg
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Road surface classification visualization showing a road surface with a graphic overlay illustrating the prediction made by an ML model. Predictions are visualized as colours corresponding to specific road surface conditions.

Typeset in LaTeX
Gothenburg, Sweden 2024

Evaluating Incremental Machine Learning Models for Road Condition Classification
David Svantesson & Julia Hansen
Department of Computer Science and Engineering
Chalmers University of Technology | University of Gothenburg

Abstract

This project was provided by Klimator AB with the aim of evaluating different incremental Machine Learning (ML) models predicting Road Surface Conditions (RSC), using a given data set from Klimator AB. Successfully identifying the RSC is associated with an autonomous vehicle's traffic safety and therefore an important area of investigation. This project presents the evaluation of seven models, with three of the models, Gaussian Naive Bayes, Complement Naive Bayes and Hoeffding Tree Classifier, being of an incremental nature in their implementation, and the remaining, Decision Tree Classifier, K-Nearest Neighbors, Logistic Regression and Dummy Classifier, employed as ensembles in an incremental manner. The models were evaluated against a Random Forest model serving as a top-level baseline, and a single Dummy Classifier serving as a low-level baseline. All models were trained using datasets derived from vehicle rides under varying RSCs, which were split into smaller intervals called epochs. The findings of this study are that the ensembles of Logistic Regression and Decision Tree Classifier demonstrate the greatest overall strengths, achieving the highest average accuracy. Additionally, the Hoeffding Tree Classifier performs strongest during rapid changes of RSC, so-called *concept drift-events*. However, the performances of all models fall short of optimal in terms of making accurate predictions. To optimize the top three candidates further, this project identifies opportunities for further development and enhancements, potentially leading to an ML model suited to assist in autonomous vehicles and ensuring traffic safety.

Keywords: Machine Learning, Incremental Learning, Road Surface Conditions, Ensemble Method, Concept Drift, Traffic Safety, Model Evaluation, Autonomous Vehicles.

Acknowledgements

This project was provided by the company Klimator AB with the purpose of evaluating different machine learning models and identifying their strengths and weaknesses when presented with RSC data. Klimator is a Gotheburg-based company that makes road weather intelligence available through predictive and detective data, aiming to empower safe, sustainable and autonomous driving [1]. The results of this report will serve as a foundation for their future work on RSC classification. David Svantesson and Julia Hansen, the authors and executors of this project would like to extend their gratitude to Klimator AB. Especially to Pontus Andersson who served as supervisor and mentor from the company, and Gustav Gulliksson for giving us the opportunity of collaboration. The authors would further like to extend their gratitude towards the Chalmers supervisor connected to this project, Adam Breitholtz.

David Svantesson & Julia Hansen, Gothenburg, May 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CNN	Convolutional Neural Network
CNB	Complement Naive Bayes
DT	Decision Trees
E	Epoch
E-DT	Ensemble of Decision Trees
E-Dummy	Ensemble of Dummy Classifiers
E-KNN	Ensemble of K-Nearest Neighbours
E-LRC	Ensemble of Logistic Regression Classifiers
ENet	Efficient Neural Network
FN	False Negative
FP	False Positive
GNB	Gaussian Naive Bayes
HTC	Hoeffding Tree Classifier
KNN	K-Nearest Neighbours
LRC	Logistic Regression Classifier
ML	Machine Learning
NBA	Naive Bayes Adaptive
NB	Naive Bayes
NN	Neural Network
RF	Random Forest
RSC	Road Surface Conditions
S-Dummy	Single Dummy Classifier
SEA	Streaming Ensemble Algorithm
SS	Subset
TN	True Negative
TP	True Positive

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

c	Indices representing a certain combination from the group of hyperparameter combinations (Total *Z-score* formula)
i	Index of a class (*softmax* function in *LRC*)
i_1,i_2	Indices for data points (*distance* formula in *KNN*)
j	Indices for data points (*softmax* function in *LRC*)
new_1, new_2	Indices for data points (*distance* formula in*KNN*)
Tot	Indices indicating the total number N of data points, used in the *balanced accuracy* formula
Tot_B	Indices indicating metrics belonging to the *balanced accuracy* formula

Sets

C	Group of hyperparameter combinations (Total *Z-score* formula)

Parameters

β	Learning parameter for *LRC*
δ	A constant, used in the *Hoeffding bound* formula
R	Range of random variables r, used in the *Hoeffding bound* formula
$\boldsymbol{\omega}$	Learning parameter for *LRC*

Variables

a	Average accuracy during one session (*Z-score* formula)
C	A class used in the *balanced accuracy* formula
ε	The Hoeffding bound
k	Total number of classes (*softmax* function in *LRC*)
K	Number of classes (*balanced accuracy* formula)
m	Number of sessions (*Z-score* formula)
μ	Mean of accuracy for all hyperparameter combinations during a session (*Z-score* formula)
n	Total number of predictions (*Accuracy* formula)
N	A number of variables or instances, used in the *Hoeffding bound* formula and the *balanced accuracy* formula
p	A variable used in the *balanced accuracy* formula
$\overline{r}$	Mean value of random variables (*Hoeffding bound* formula)
σ	Standard deviation in accuracy during a session for all hyperparameter combinations (*Z-score* formula)
x	A vector of a data point (formulas of *LRC* and *KNN*)
$\mathbf{X}$	A data point (formulas for *Naive Bayes models*)
y	Denotes either the true class of a data point or a vector storing the true classes of data points (*Accuracy* formula)
$\hat{y}$	Predicted class of a data point (*Accuracy* formula)
Y	A class (formulas for *Naive Bayes models*)
z	Dot product between $\boldsymbol{\omega}$ and a vector of a data point (formulas of *LRC*)
Z	Z-score

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

Detecting the road surface conditions (RSC) is an important feature of an autonomous vehicle. RSC is associated with the vehicle's traffic safety as it influences the vehicle's braking control due to variations in friction [2]. The US department of transportation reported in 2017 that 22 percent of vehicle accidents were weather-related and that the majority of them occurred during wet, snowy or sleet road conditions [3]. The need for rapid and accurate categorisation of the RSC is therefore a necessity for an autonomous vehicle to be able to operate safely and efficiently.

Machine Learning (ML) can be used for assessing RSC. ML is a technique which is used for categorizing data [4]. For example, classifying objects based on image data, laser measurements or similar technology [2]. For instance, ML is employed in detecting brain tumours on magnetic resonance images and provides valuable information for accurate diagnosis [5]. This degree project will focus on exploring different ML models and their suitability for categorizing weather-related RSC, such as the occurrence of snow, water and so on. This will be done utilizing data captured from a camera and two lasers mounted on a vehicle. These models could improve the safety of autonomous vehicles and potentially be of assistance in non-autonomous vehicles.

1.2 Purpose

To better comprehend and ascertain the level of accuracy that can be achieved in RSC categorization, this project will utilize a dataset provided by Klimator to explore incremental ML models and evaluate their performance. The results from this project will serve as a basis for comparison of existing and possible future implementation of RSC categorization ML models within the company Klimator.

1.3 Goals

- Evaluate and assess the performance of different ML models on different RSC data, using evaluation metrics conventional to the field of study.
- Based on the evaluation results, discover and identify the strengths and weaknesses of the models in relation to the RSC data, with consideration given to traffic safety.
- Provide Klimator with a foundation for further research and development, supported by scientifically grounded methodologies and the project results.

1.4 Limitations

This degree project will explore different existing ML models specifically for the categorization of RSC. Categorization of other objects is not a part of this project. The project treats each category equally and does not consider the consequences of misclassifying certain categories. Therefore, no higher importance is given to categories that would pose a higher risk if misclassified, during training of the models. In accordance, the project does not take into account the impact the different categories should or may have on decision-making in, for example, an autonomous vehicle. Additionally, the aim is not to find an optimal model but to explore different models and approaches, and compare them with one another to serve as a basis for recommendations to Klimator regarding future research. Likewise, the aim is not to create software able to be used by an autonomous vehicle in a real-life scenario. The software used for evaluating models in this project emulates a real-life scenario, it does not regard or encompass all its elements. The project does not involve collecting data and relies solely on the data provided by Klimator.

2

Theory

This chapter covers theory related to the work of this project. The purpose is to provide foundational knowledge to facilitate understanding of the project's methodology and evaluation process. This chapter includes basic knowledge of machine learning, incremental learning, and models that concern this project. In addition, the chapter comprises both key considerations when modelling a machine learning problem, as well as evaluation metrics to use during the evaluation procedure. Finally, it accounts for the principle of concept drift - a problem specific to incremental learning.

2.1 Machine Learning

Machine learning is a subfield of artificial intelligence which focuses on using algorithms and data to enable machines to demonstrate intelligent behaviour. It is used, among other things, for categorizing and predicting outcomes. Machine learning can be simply explained as an algorithm that processes data, such as texts or images, and learns to recognize patterns. The processing is referred to as "training" the model. There are several types of learning processes; two of them are *supervised* and *unsupervised*. Supervised learning indicates that there is a correct answer that the model should learn for each piece of data. The training set could be handwritten text and the goal could be to create a model for writing recognition. Unsupervised learning is used to partition data into sets, clustering data together based on shared commonalities [4].

2.1.1 Data Representation in Machine Learning

Independent of whether supervised or unsupervised ML is used, the ML algorithm involves processing data. Data is stored in a dataset. The dataset includes input variables, that store *'feature values'* which describe the observed data. A data point is one unique observation in the dataset with its associated feature values. In the case of supervised learning, the data point also contains an output variable that stores the true value associated with the observation. The true value could be the category the data point belongs to [4].

Variables can store either *numerical* or *categorical* values. A value is considered numerical if there is a natural ordering, whereas it is considered categorical if the value belongs to one of a finite number of predetermined categories. In supervised machine learning, the problem is either a Classification or Regression problem, depending on whether the output value is categorical or numerical [4].

2.1.2 Incremental Learning

Many ML algorithms are designed to train on data points retrieved as one batch and the model is never re-trained afterwards [4]. Incremental learning, however, is a machine-learning strategy where the algorithm continuously adapts and updates its model parameters as new data arrive on a data stream [6, 7]. Data points are received sequentially and processed either one at a time or as chunks of data [7]. This strategy allows the model to adapt to changes and evolve over time without the need for rebuilding from scratch. This feature enables real-time adaption and reduces the need for big training sets [6, 7].

2.2 Machine Learning Models

There are both incremental and non-incremental models available in supervised ML. Incremental models permit incremental learning whereas non-incremental models do not. However, some models can handle both approaches [8, 9]. The following sections address models that are utilized in this study and account for theories behind using non-incremental models in an incremental fashion.

2.2.1 Neural Networks

A *Neural Network* (NN) is an ML model, traditionally trained once, without support for further incremental learning. Such a network consists of a few or several layers. Each layer contains interconnected nodes, where each connection has an associated weight. The weight determines the importance of the connection and can be adjusted during the training process. Each node also has a threshold value that can determine when a node is activated, meaning when it's sending data to its connected nodes in the next layer. The first layer, called the input layer is where data enters the network. Depending on the depth of the network there are one or multiple hidden layers and finally one output layer. The output layer represents the final outcome of the network. In a classification network, the outcome could be the probability that a given input belongs to a pre-defined category [4].

2.2.1.1 Convolutional Neural Networks

A *Convolutional Neural Network* (CNN) is a specific type of neural network often used for image, video, and pattern recognition. The network is a multi-layer network based on matrix multiplication and linear algebra to identify patterns within an image. CNN uses a filter to analyze and identify different parts of the picture one at a time which makes the network efficient when processing images and video [4].

Furthermore, CNN can also have an encoder-decoder architecture. Encoders compress image input, filtering so-called *noise*, which refers to information that does not contribute to the categorisation of a group of pixels. The decoder expands the dimensions of the input it receives from the encoder, and the nature of the output varies according to the network's objective [4]. An example of such a network is the Efficient Neural Network (ENet), where the output signifies pixel-wise probabilities for a specified set of output classes [10].

2.2.2 k-Nearest Neighbors

K-Nearest Neighbours (KNN) is a model that makes predictions based on proximity and is typically trained in a non-incremental fashion. The model compares the features of a given data point with those of the training data points that are closely located in the data space. The data space can be either one-dimensional or multidimensional, depending on the number of features the data points consist of. In a classification problem, to predict the class of a test data point, the model identifies the 'k' nearest neighbours in distance to the test point. These training points are then used as a basis for predicting the class of the test data point, where each training point votes for its own class [4]. KNN can be configured to weigh these neighbours' votes differently based on their distance to the test data point, or to give them equal weights [11]. The class with the most votes wins and serves as the prediction of the test point [4].

The simplest possible model is the 1-nearest neighbour where 'k' is set to 1. However, using a higher 'k'-value is more common since 'k' = 1 creates a less reliable model. The models can use different algorithms to calculate the distance to the neighbours. The default method is the Euclidean distance algorithm. For a new data point to be predicted $\mathbf{x}_{\text{new}}$ and an already predicted data point $\mathbf{x}_i$ in a two-dimensional space, the distance between them is calculated using the following formula [4]:

$$distance = \sqrt{(\mathbf{x}_{\text{i1}} - \mathbf{x}_{\text{new1}})^2 + (\mathbf{x}_{i2} - \mathbf{x}_{\text{new2}})^2} \tag{2.1}$$

2.2.3 Logistic Regression Classifier

Another model which traditionally is not trained in an incremental fashion is the *Logistic Regression Classifier* (LRC) [12]. In the binary classification case, where there exists a positive and a negative class, the classifier computes the probability of a data point belonging to the positive class using a *logisitc function* [4]:

$$P(Y = positive|z) = \frac{e^z}{1 + e^z} \tag{2.2}$$

Where,

$$z = \beta + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_p x_p. \tag{2.3}$$

β and $\boldsymbol{\omega}$ are so called *learning parameters* that the model learns during training. z is hence the dot product between the vector $\boldsymbol{\omega}$ and the feature vector $\mathbf{x}$ of a data point, with the addition of β. The probability of the negative class is then [4]:

$$P(Y = negative|z) = 1 - P(Y = positive|z), \tag{2.4}$$

since the sum of all probabilities is equal to 1. For a multiclass problem, one can either train one classifier per class using a one-versus-all approach [12], or compute the probability of class i using the so-called *Softmax function* [4]:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}, \tag{2.5}$$

where k is the total number of classes. Because there is one z_i per class $i = 1, ..., k$ for a given feature vector, each class has its own $\boldsymbol{\omega_i}$ and β_i in the multiclass model. In other words, each class has its own learnable parameters which are learned during training [4].

Training of a LRC aims to find the values of $\boldsymbol{\omega_i}$ and β_i for each class $i = 1, ..., k$, which minimizes a loss function. The loss function for the multiclass problem is referred to as the *cross-entropy loss* and has the following formula:

$$J(\boldsymbol{\omega}, \boldsymbol{\beta}) = -\frac{1}{n} \sum_{j=1}^{n} \ln \Big(\text{softmax}(z_{y_j}) \Big) \tag{2.6}$$

where $\boldsymbol{\omega} = (\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_k)$, $\boldsymbol{\beta} = (\beta_1, \ldots \beta_k)$, n is the total number of data points observed and $softmax(z_{y_j})$ is the probability of the model predicting the correct class y given the feature values of the data point j.

There are different algorithms available for optimizing the loss function [12]. In addition, a *regularization parameter* can be added. This parameter is commonly used by many models and serves to prevent the model from learning too strong dependencies which could cause the model to perform poorly on data not seen during training (see Section 2.3 on *overfitting*) [4]. There are also different types of regularization parameters available. The effect of the parameter is regulated based on an additional parameter, referred to as the *regularization strength* [4, 12].

2.2.4 Decision Tree

There are several ML models that involve the concept of Decision Trees (DT). The basic DT model only supports non-incremental learning [13], however, there are incremental versions available [14]. A DT is a tree like, hierarchical structure of nodes, including a root node, internal nodes and leaf nodes. The structure is visualized in Figure 2.1. Each node, except for the root node, is positioned at the end of a branch coming from a node higher up in the structure. There are two branches emanating from each node and hence each node can connect up to two additional nodes. Leaf nodes are those positioned at the bottom of the tree, that do not connect any more nodes. A leaf node corresponds to a certain class. However, a class can be represented by several leaf nodes, depending on how the DT is built. When using a DT to classify a data point, the data point's feature values are evaluated against conditions in the tree. Both the root node and the internal nodes are associated with one condition each. First, the feature value corresponding to the condition of the root node is evaluated. How this feature value relates to the condition determines whether the condition of the node located at the end of the left or right branch is evaluated next. The process continues in such a manner, evaluating one condition at a time, travelling down the tree until a leaf node is reached, predicting the class of the data point [4].

Likewise, the process of building a decision tree starts at the top by creating a root node. The goal is to select a value for one of the features to use as a condition and choose it such that evaluating data points against that condition results in the purest split. The essence of the purest split is to partition data points into two subsets, with the goal of achieving maximal homogeneity within each subset by harbouring data points belonging

to the same class. This is done by, for each of the features, iterating through different splitting values, optimizing a *splitting criterion*. There are different splitting criteria available when training a DT classifier, however, all of them strive to estimate the homogeneity within the subgroups formed. The feature with the optimal splitting value is selected to serve as a *splitting condition* for the node. The condition divides the data points into two subsets, serving as two leaf nodes. The algorithm then proceeds, determining the splitting conditions for the two leaf nodes in a similar fashion, resulting in additional leaf nodes. This process continues until a stopping criterion is fulfilled. This stopping criterion varies depending on implementation, and one or several criteria may be defined. Common scenarios terminating further expansion of the tree involve reaching a set maximum depth, or when a further split would result in too few instances within a leaf node [4].

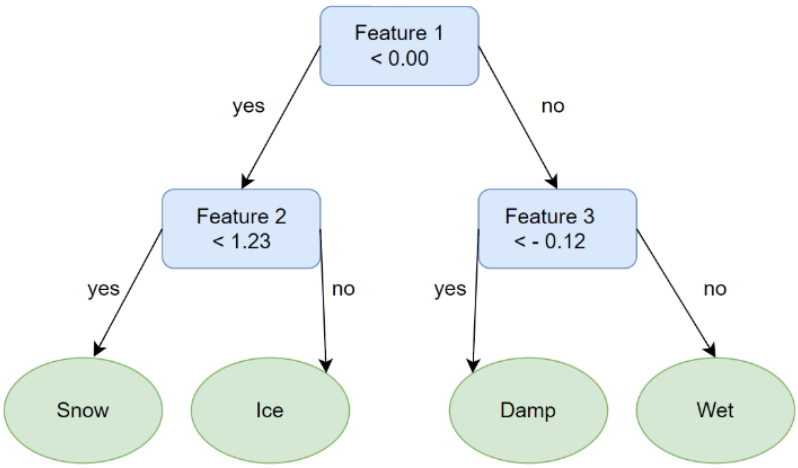


Figure 2.1: Decision Tree structure. Visualizing a decision tree structure with splitting conditions (blue boxes) and leaves (green ovals). Feature values of a data point are evaluated against splitting conditions, starting from the top of the tree, evaluating splitting conditions until reaching a leaf. The final prediction of the tree is contained within the leaf.

2.2.4.1 Hoeffding Tree Classifier

Hoeffding Tree Classifier (HTC) [14] is a decision tree-variant, where the tree is built step by step through processing data incrementally. The concept allows a decision tree to be built, without the necessity of storing already processed data. The core of the algorithm is to minimize the number of instances used to determine which attribute that is optimal to use as a splitting attribute while maintaining a high level of accuracy in the attribute selection. The number of instances is determined using the so-called *Hoeffding bound*:

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2N}} \tag{2.7}$$

The theorem is based on the assumption that N random variables r in the range R are observed independently. δ is defined by $(1 - \delta)$ being the probability of the true mean value of such a random variable being at least $\overline{r} - \varepsilon$. In the HTC, ε defines a minimum difference between the split evaluation value of two attributes. These values are determined by a heuristic function G, which estimates the splitting property of each attribute after observing N instances [14].

As for an ordinary decision tree, the HTC starts with a single leaf node. For each instance processed, the split evaluation value of each attribute is computed. If the difference between the attribute deemed as resulting in the best split and the one resulting in the second best split is greater than *epsilon*, enough instances have been processed to convert the leaf node into a node with a splitting criterion and two leaf nodes as children. Initially, the single leaf node is thus converted to a root node. Subsequently, each processed instance is sorted into one of the leaf nodes based on how its value of the relevant attribute compares to the splitting criterion value. As more instances are processed, leaf nodes are converted into nodes with splitting criteria, and additional leaf nodes are formed [14].

The HTC is thus built incrementally, increasing in size. As the tree grows, more and more leaf nodes emerge, each node corresponding to the classification of a class. Growing the tree will alter the proportions of the classification of the classes, which makes it possible for the tree to adapt as new data comes in. However, when a node with a splitting criterion has been formed, it stays in the tree permanently. Thus, the tree can merely grow but not be rebuilt [14]. Eventually, the tree will reach a size where the model starts to *overfit* on training data. Overfitting is when a model memorizes patterns in training data which leads to a good performance when tested on training data but a poorer performance on test data, see Section 2.3. Different regularization schemes has been proposed to overcoming the issue of unnecessary growing and overfitting of the HTC, such as limiting features available when a split is being considered [15]. The Hoeffding tree algorithm also assumes that the distribution in the dataset remains stable during the incremental training process [14].

2.2.5 Naive Bayes Models

Given that incremental training involves adjusting predictions based on new data while retaining knowledge gained from previous observations, *Bayes' Theorem* [4] serves as a foundational principle for such a machine learning model. Mathematically, the theorem is stated as follows:

$$P(Y|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|Y) \cdot P(Y)}{P(\boldsymbol{X})} \tag{2.8}$$

This formula takes into account the probability of a class Y, $P(Y)$, before observing new data, denoted by $\boldsymbol{X}$. $P(Y)$ is referred to as a *prior* while the calculated probability $P(Y|\boldsymbol{X})$ is referred to as the *posterior*. The formula allows for an incremental update of the posterior by using the former posterior as the new prior in the next iteration. In the context of a machine learning classification problem, $\boldsymbol{X}$ represents a set of values, such as feature values of a data point, whereas Y denotes a class label. Thus, the formula can be utilized to calculate the probability of a class label, given a former probability of the class and a set of observed feature values [4].

There are several variants of models utilizing Bayes' Theorem, all referred to as *Naive Bayes* (NB) models. Examples of such models include *Gaussian* NB [8] and *Complement* NB [9]. Both of these models support incremental learning.

2.2.5.1 Gaussian Naive Bayes

A variant of a ML model that is based on Bayes' Theorem, is *Gaussian Naive Bayes* (GNB). This variant calculates the posterior probability under the assumption that values of $\boldsymbol{X}$ are continuous and follow a normal distribution. Additionally, features are presumed to be independent from one another. A data point is predicted as the class which have the highest calculated posterior probability under these assumptions [16].

2.2.5.2 Complement Naive Bayes

Another ML model based on the Bayes' Theorem, is *Complement Naive Bayes* (CNB). The Complement Naive Bayes is particularly well suited to handle imbalanced data sets [9]. For each class, CNB computes the posterior probability based on how training data relates to the other classes, excluding the class currently investigated. These other classes, used for computing posterior probabilities, are referred to as complementary classes. This strategy creates less biased estimates of the *posterior* since a more even amount of training data per class is being used, minimizing bias towards majority classes [17].

2.2.6 Ensemble Methods

An ensemble method is a method where several instances of some, or several ML models are combined and used together. The group of models is called an ensemble. The purpose of an ensemble is to harness the "collective wisdom" of the models. This approach is based on each of the models in the ensemble being trained to recognize different patterns within data. Two common methods when creating ensembles using stationary data for training are *bagging* and *boosting* [4]. In *bagging* several models are trained, each on a different subset of data, sampled from an original data set. *Boosting*, on the other hand, incorporates models incrementally, by each iteration letting the errors of the current ensemble's predictions dictate the focus for the next model to train [4]. When training an ensemble of models on data accessed through a data stream, there are several methods available [7]. If using a *chunk based approach*, a new model is trained on the next batch of training data. It is then added to the ensemble, potentially replacing one of the other models proven less successful [18, 7]. A data stream ensemble can also be combined with the method of *bagging* or *boosting* [7]. Regardless of the method used to construct an ensemble, each model in the ensemble generates predictions independently, which are then combined to obtain a final prediction. The benefit of this approach is to reduce the variance of our estimates [4].

There are different approaches on how to decide on the prediction of the ensemble. A basic approach is to let the prediction be determined by the *majority vote*. In this approach, each model's prediction is regarded as a vote, and the category with the most votes is selected as the final prediction [4]. In the case of a tie, the category could be chosen arbitrarily. A slightly more advanced approach is to instead use a *weighted* majority vote to cast a prediction. In such an approach, each model's prediction is assigned a weight which determines how much influence the model's vote has on the final prediction. There have been several proposed mechanisms for determining the weights of each ensemble model's vote [7]. Such mechanisms often involve assigning weights according to how each model performs in comparison to the other models of the ensemble [7], for example by adding to the weight of a model when making a correct prediction, adding more weight when fewer of the other models get the prediction right [19]. Using a weighted majority vote can improve an ensemble's overall performance [7].

2.2.6.1 Random Forest

A *Random Forest* (RF) is built upon the concept of bagging. The method constructs an ensemble of decision trees, creating each tree from an individual subset sampled from the original data set. However, the technique introduces additional randomness by considering only a subset of features when deciding on splitting a node, employing different subsets of features for different nodes. In addition, each tree employs its own set of such feature subsets. As a general guideline, the number of features in the subset is equal to the square root of the total number of features [4].

The advantage of an RF model lies in its capacity to enhance generalization. Bagging itself reduces the risk of model overfitting, meaning that it prevents the model from finding patterns in training data that are not present in the intended population, see Section 2.3. This is done by decreasing the variance of predictions, thereby minimizing the risk of the model memorizing noise-induced patterns. The risk of overfitting is reduced even further due to the RF constraint on the number of features considered at each node, since

the features resulting in the best split are not necessarily considered. This makes the RF model equipped to generalize well on new data [4].

2.3 Modeling a Machine Learning Problem: Key Considerations

Which type of model that is optimal in a given situation, depends on the constitution of the dataset and the complexity of patterns in the data. It is common to try different types of ML models, starting with a more basic type which can serve as a baseline, before trying a more complex type to train. In the case of classifying pixels in images, it is common to use features extracted from the images by a pre-trained neural network [4].

How well an ML model can be adapted to patterns in training data depends on a model's complexity. The higher the complexity, the more complex patterns can be modelled. Data containing more complex patterns should therefore benefit from a more complex model, with the potential of finding such patterns. Different models can also find different patterns. For instance, KNN favours similarity between data points, treating all features equally. In contrast, a decision tree assigns different importance to features, and depending on the splitting value of a node, two data points with feature values close in proximity could therefore be classified differently. A neural network, on the other hand, has the potential to find more abstract patterns in data. However, a model's complexity also varies depending on parameter settings. KNN gets more complex, the lower *'k'*-value that is used and a neural network increases in complexity the more layers that are added. However, finding more complex patterns is not always a sought-after property. Such patterns could be due to *noise*, meaning data points that are not representative of any of the possible classes. This phenomenon is called *overfitting*. Overfitting leads to the model performing well when predicting on training data, but poorly when predicting on not previously seen data. Therefore, a more complex model is not necessarily optimal, as it increases the risk of overfitting. On the other hand, too little complexity in a model can result in an overall poor performance since it may fail to capture true patterns [4]. In addition, there is a trade-off between the complexity of a model and its efficiency, since more complex models tend to use more computational resources and take a longer time to train.

2.4 Evaluating a Machine Learning Model: Performance Metrics

After training a model, its performance can be evaluated using certain *performance metrics*. Several different metrics exist, each providing a unique insight into the model's performance. The formulas for the different metrics are based on the classification outcome categories: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The group of correct predictions can be ordered into the subgroups TP and TN. TP is the group where a model correctly predicted an affiliation and TN is the group where a model correctly predicted a non-belonging. The incorrect predictions can be ordered into FP and FN. FP is the group where a model predicts an affiliation but there is none, and FN is the group where a model predicts a non-belonging but there

is an affiliation. The four groups combined represent all the possible outcomes [4]. The following metrics are some of the most commonly used.

2.4.1 Accuracy

The Accuracy metric evaluates the model by measuring the proportion of correct predictions compared to the total number of predictions. A high value indicates that the model's predictions are mostly accurate. The accuracy value is calculated according to the following formula [20].

$$\text{Accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{1}(\hat{y}_i = y_i) \tag{2.9}$$

The vectors y and $\hat{y}$ represent the true values and the model's predictions respectively, while n denotes the total number of predictions made by the model.

Another formula used for calculating accuracy is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.10}$$

This formula can be used for binary classification, where there exists a positive and a negative class [4]. It can also be utilized in multiclass classification, for calculating the model's accuracy in predicting individual classes.

2.4.2 Recall

Recall is a metric that describes how large the proportion of the TP group is, among all actual positive instances. Since the model can falsely predict that there isn't an affiliation, it is interesting to know how frequently it predicts correctly. This metric is useful when minimizing the proportion of FN is important. A high value on recall indicates that there is a large proportion of TP cases and low FN cases [4].

$$Recall = \frac{TP}{TP + FN} \tag{2.11}$$

2.4.3 Precision

Precision is a metric describing how large the proportion of TP is of all positive predictions. Since the model can falsely predict an affiliation (FP) it is interesting to know how often that happens. The larger the value, the larger the precision of the model [4].

$$Precision = \frac{TP}{TP + FP} \tag{2.12}$$

2.4.4 F1-score

F1-score combines the metrics of recall and precision. A high F1 score indicates that the model has both high recall and high precision. F1-score is commonly used when there is an imbalance in data. If one of the categorization classes is more common than the others, a model that often falsely predicts affiliation to that class can incorrectly be regarded as accurate. The F1-score can provide a comprehensive assessment of how well the model performs across all classes [4].

$$F1\text{-}score = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.13}$$

2.5 Concept Drift In Incremental Learning

When evaluating a model, it is important to consider what happens to the data over time and what might affect the model's performance. In a dynamically changing and evolving data stream, the data distribution can change over time. This phenomenon is called concept drift [21]. Traditional non-incremental learning methods require a representative data set to be available before the training. When such a model trains on a specific type of data distribution, it expects to predict based on the same data distribution [4]. The model's performance can be affected by classes being over or underrepresented in the data. So if the data during prediction has a different distribution - i.e., concept drift has occurred and a previously underrepresented class becomes more prevalent, the model may experience degraded performance [22]. A relevant example of when concept drift can occur is during a real-time data stream of RSC. This data is ever-changing due to seasonal and periodic changes in the weather, such as rain, snow, ice and slush.

The problem for a machine learning model when the data drifts, is that recently learned rules of prediction lose effectiveness. Unlike non-incremental models, which tend to degrade over time and fail to ensure high predictive performance [23], incremental models are better suited to handle changing data because of their ability to evolve and adapt [21]. However, even with incremental and adaptive learning algorithms to handle concept drift, more research is needed regarding their robustness, reliability and scalability [24]. Several algorithms for handling concept drift, including multiple ensemble models, hold disadvantages of varying degrees, such as failing to predict unbalanced data flow well [25]. Difficulties also exist for an ML model to distinguish between noise in the data and a concept change, further complicating the issue [26].

3
Methods

This chapter describes the project methodology. It outlines the systematic approach adopted by this project, based on the theory described in the former chapter. Methods are presented in a logical rather than strictly chronological order. This is to keep the level of interpretability high, as certain steps, such as training different models, follow a similar approach, thereby making it reasonable to deviate from a strictly chronological presentation.

3.1 Utilized dataset

This project used a pre-collected dataset consisting of laser measurements and images taken on different Road Surface Conditions (RSC) during vehicle rides, captured during varying times, lightning, and weather. The rides had been split into sessions lasting approximately 10 minutes, with data points ranging from 6 400 to 150 000, on average about 95 900. The dataset contains a total of 20 sessions, with a total of 2 013 420 data points. For specific information on data distribution see Section A.2.1 in the Appendix. During these sessions, two lasers were measuring at two points beneath the vehicle whilst a camera took forward-facing images. In the images, the laser point's location can be identified, providing a visual representation of each measured point, as seen in Figure 3.1. Each data point entry in the dataset contains a timestamp, a reference to the related image, and the following measurements from the laser: coordinates of the laser point related to the vehicle, probabilities of dry, damp, wet, snowy, slushy, and icy RSC, the determined RSC category, and also the estimated friction and water-depth. Furthermore, each entry contains four features extracted from an ENet network's processing of the respective image. The properties of the images had been processed into quantifiable data, enabling various ML models to utilize them as input features. This project considered the RSC category determined by the lasers as the true RSC. The RSC categories of interest are dry, damp, wet, snow, slush, and grey ice. Their distribution can be seen in Table 3.1.

RCS	**Grey ice**	**Slush**	**Snow**	**Wet**	**Damp**	**Dry**	**Total**
Data Points	347 216	35 002	558 633	203 123	566 649	302 797	2 013 420
Percentage %	17.25%	1.74%	27.75%	10.08%	28.14%	15.04%	100.0%

Table 3.1: Data points distribution. Showing the distribution of the data points and percentage by RCS condition

Figure 3.1: RSC Determined by the Lasers During a Vehicle Ride. Showing coloured dots representing the points on the road surface where the lasers have performed measurements. For each measured point, the colour corresponds to the RSC determined by the laser. The legend in the top left corner describes the specific RSC categories and their respective colour.

3.1.1 Preprocessing of the Dataset

Before being utilised by the ML models, the session data was sorted to only include relevant information. The following information was kept for each data point: The timestamp, the related image reference, the laser point coordinates, the four features and the determined RSC category. For an example of data points, see Table A.2 in the Appendix. Each session was sorted after the variable timestamp in chronological order to be able to simulate a continuous vehicle ride. The data points categorized as belonging to the categories of interest were extracted, while those not belonging were discarded. The categories of interest were a limitation determined by Klimator. The timestamp format was converted into a format that allowed for numerical operations, which was necessary to partition the data based on time, see Section 3.4. For one of the models, Complement Naive Bayes, see Section 3.3, the features were scaled to fit the interval 0 to 1, since the model was unable to handle features of negative values [9]. The scaling was conducted by using MinMaxScaler from sci-kit learn preprocessing library [27].

3.2 Utilized Software and Libraries

To implement the machine learning process this project utilized the programming language Python [28]. Libraries such as NumPy [29], and Pandas [30] were used to handle computations, create and manipulate data frames and save the results externally. Functions for calculating accuracy, precision, recall and f1-score, originated from the library sci-kit learn metrics [31]. Further functions utilized for calculating metrics were a creation of this project and are described in Section 3.6.5. Matplotlib [32] was used for visualizing results as plots.

3.3 Project Models

This project evaluated multiple models. These models can be arranged into three groups. The *standalone models*, the *ensembles* and the *baseline models*. The standalone models are the incremental models Gaussian Naive Bayes and Complement Naive Bayes that both originate from sci-kit learn [8, 9], and the Hoeffding Tree Classifier, originating from sci-kit multiflow [33]. The ensembles are models constituting an ensemble of classifiers. One ensemble for each of the following classifiers was constructed: Decision Tree Classifier, K-Nearest Neighbors, Logistic Regression and Dummy Classifier, all originating from sci-kit learn [13, 34, 12, 35]. The ensembles of these classifiers will be referred to as E-DT, E-KNN, E-LRC and E-Dummy, respectively. Ensemble models were used since they enable non-incremental models to be used in an incremental fashion and are a common approach when dealing with data streams [7]. The last group of models called "the baseline models", were utilized to create benchmarks for comparison. The specific models used for this purpose consisted of the Random Forest Classifier and a Dummy Classifier, both types of classifiers originating from sci-kit learn [36, 35]. The Dummy Classifier used for creating a low-level baseline will be referred to as the Single Dummy Classifier or S-Dummy, see more in Section 3.3.2. Another use case for the Dummy Classifier was also as a helping classifier in the Logistic Regression ensemble. Since the Logistic Regression classifier requires instances of at least two classes to train, a Dummy Classifier was trained and added to the ensemble instead, in cases where the training set consisted solely of instances belonging to one class. The Dummy Classifier was trained to always predict the most frequent class of the data points in the training set, in all use cases. Hence, when used in the Logistic Regression ensemble, it was set to predict the only class present in the training set.

3.3.1 Random Forest as a High-Level Baseline

To be able to benchmark and assess the different models evaluated by this project, an attempt to achieve a high-level baseline was made by utilizing a Random Forest model. Random Forest is a relevant option to accomplish this, given its high performance in various fields [37, 38, 39, 40]. Additionally, by using a large number of trees, (100 or larger) the classification error can drop significantly, compared to just using a few [41]. The RF model was employed solely as a baseline model due to the significant increase in computational power required when using a large number of trees [4]. Thus making the model unfit for real-time prediction of RSC. The RF was trained according to the process described in Section 3.4.3.

3.3.2 Single Dummy Classifier as a Low-Level Baseline

To further assess the performance of the different models, this project implemented a straightforward model by utilizing the Dummy Classifier [35]. A Dummy Classifier does not learn from the data or consider data patterns but simply predicts labels according to the data distribution of the training set [42]. If category 1 was most frequent in the training set, the model will classify everything as 1 [35]. The approach involved using a single Dummy Classifier training on a subset, predicting on the next, and continuously being replaced by a new classifier each epoch. This resulted in a model always predicting the class being in the majority in the previous epoch. Using a Dummy Classifier is a strategy

for getting a baseline to evaluate more sophisticated models. The expected outcome of the Single Dummy Classifier is a simple baseline representing an unsophisticated model's performance. It serves as a benchmark indicating the minimal performance expected from the models evaluated by this project.

3.4 Training a Model

The ML models in this project either use incremental learning as an individual classifier or operate as an ensemble of classifiers. The different variants have been trained slightly differently as described in the sections below. Common settings for all approaches are how the data is partitioned. The subset of a dataset used for training is called the *training set*. Data used for testing is hence called *test set* [6]. The provided dataset consisted of multiple measure sessions. All the data had been collected in advance and was available simultaneously, separated into sessions. Each session was collected during a ride with a vehicle equipped with a camera and two lasers. As seen in Figure 3.2, a real-time scenario was simulated with data arriving in a continuous stream using a timestamp "T". The scenario would correspond to a vehicle driving forward, with the model continuously updating as new data streams in. The timestamp T would be considered as the current time. The data before the timestamp T would be considered as "seen" data, serving as training sets. The data after T and within a given range, would be regarded as "unseen" data. Meaning, not yet used for training the model. This data would serve as test sets. The data consisted of camera images and laser measurements. The models were trained on the features extracted from the ENet processing the camera images. They predicted on the images showing the current and forthcoming view in front of the car at the timestamp T. The learning process was supervised, which means that the models were trained using labelled data, where each input is associated with a corresponding target output [4]. In this context, the laser measurements served as a true label for specific pixel groups on the images. As the vehicle moved forward, the lasers mounted underneath would classify the road ahead, providing labels for previous observations and thereby facilitating incremental learning.

The timeframe created with the use of the timestamp T is referred to as an epoch. As seen in Figure 3.2, the initial epoch refers to the timeframe beginning at the session start time (T0) and ending at the time T. The first value of T is a time called epoch-time added to T0. The value of epoch-time is set to a maximum of 10 seconds, a constraint established by Klimator due to a memory limitation. A subsequent epoch is thus the timeframe starting at the previous value of T and extending 10 seconds forward, generating the new T value, and so forth. A data set was assembled during each epoch's timeframe, consisting of the data seen during that epoch. This data set is referred to as a subset (SS). The subset could contain a varying number of data points, because of varying sampling rates and discarded samples. The overall process and the decision to create epochs based on a fixed time was a request from Klimator.
Each session was considered a separate and independent evaluation environment. This meant that the models started from scratch with zero pre-training when presented to a session. Finally, for all learning approaches, total performance and accuracy on all the sessions were calculated using various metrics.

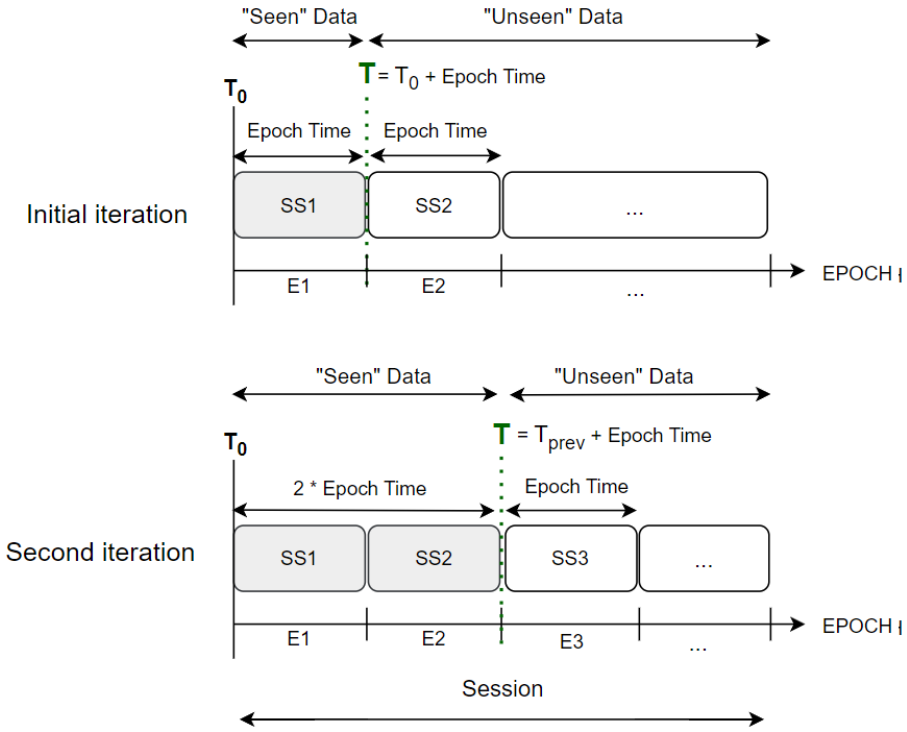


Figure 3.2: Dividing a Session into Epochs. Creating epochs (E) and subsets (SS) by utilizing the timestamp T and Epoch Time in a session.

3.4.1 Basic Training Method

As seen in Figure 3.3, T was used to divide the session into epochs. Data subsets belonging to each epoch were used as a training set (blue) and were processed incrementally, step by step. The model's predictions were for each step evaluated on data beyond time T, the test set (green), which was still "unseen". In the next iteration, T moved one epoch size forward in time and the previous test set became the new current training set, and so on. The epoch size for this procedure was set to 10 seconds.

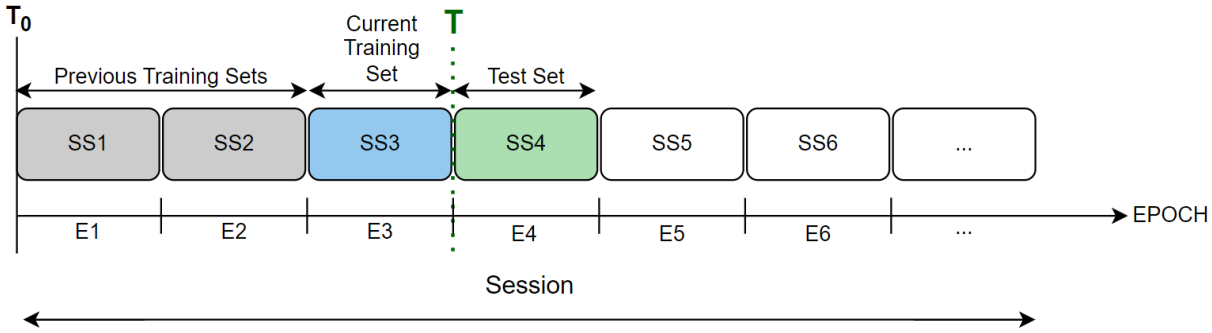


Figure 3.3: Basic Training Method. Epochs (E) in a measurement session showing incremental steps through subsets (SS) visualized at the time T.

3.4.2 Training Method Using a Sliding Window

It is not unusual for data stream-ensemble methods to utilize a *sliding window* as part of their strategy to address class imbalance and concept drift [7]. Therefore, a different training regime was developed as an alternative for the ensemble models of this project and regarded as a hyperparameter for those models, see Section 3.6.3. The sliding window of this regime abides by the basic process described by Wang and Wu [43].

As seen in Figure 3.4, T was used to divide sessions into epochs in the same way as described in Section 3.4.1. The differences in this procedure lie in the size of the training set. By utilizing a "sliding window" with the size of 5 subsets the training set grew incrementally during the first 5 epochs. During the initial iteration (epoch 1), the training set consisted of subset 1 (SS1). During the second iteration (epoch 2), the training set expanded by adding subset 2. In the following manner, the training set expanded by adding all the seen data until the sixth iteration (epoch 6). During this epoch, the first subset, SS1 was dropped. In its place, subset 6 was added, creating a “sliding window” of the training set over the session. Due to the limitation of a total of 10 seconds training set, the epoch time during this training procedure was set to 2 seconds. Following this procedure, the models generated after the first epoch get the opportunity to train on an increasing volume of data, thereby enhancing their chance of accurate predictions.

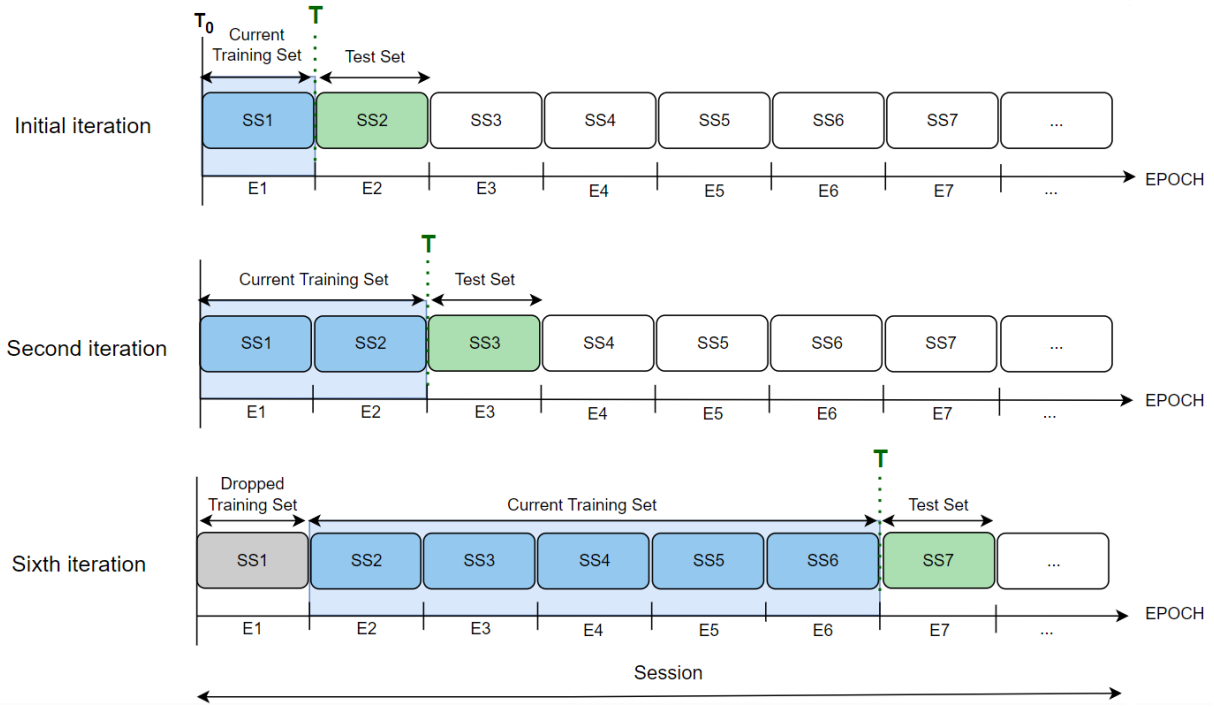


Figure 3.4: Training Method using a Sliding Window. Creating a training set using the sliding window technique. The training set will increase in size and eventually contain 5 subsets (SS).

3.4.3 Training Method of Top-Level Baseline Model

Training the Random forest followed the basic training method described in Section 3.4.1, with the following difference. As seen in Figure 3.5, during each epoch, a new RF were trained on all the previously seen data within the current session. Initially, an RF was trained on the first subset (SS1) and predicted on the following subset (SS2). Subsequently, a new RF was trained on the first subset (SS1) and also the second subset (SS2) and predicted on the third subset (SS3). During the last iteration, a final RF would be trained on all subsets preceding timestamp T. During this procedure, no memory limitations were set since the procedure is used for evaluation purposes. Using this procedure, each epoch within every session gets a baseline of assumed high performance, with multiple metrics serving as a foundation for comparison. This method to evaluate each epoch allowed for the analysis of specific epochs and time frames as intended.

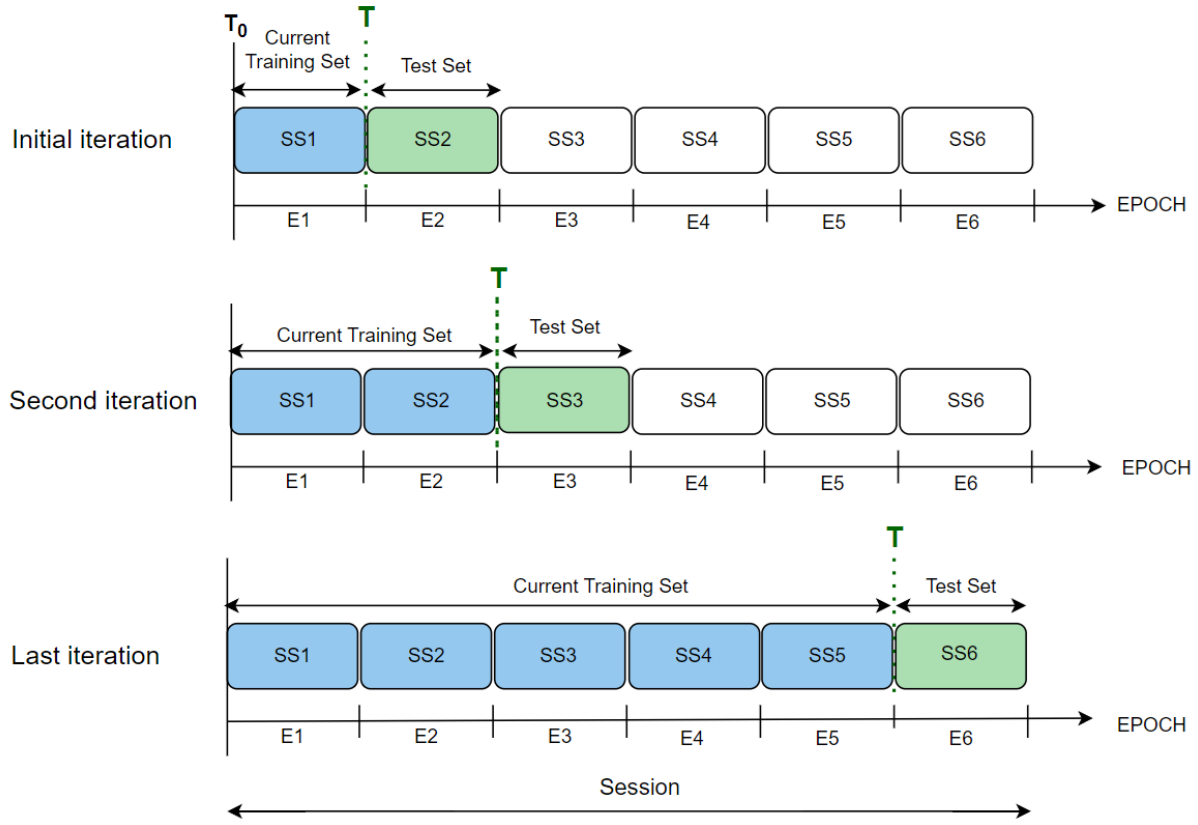


Figure 3.5: Training Method with increasing test set. Creating a progressively expanding training set of all the previously seen data. This technique is used for training the RF model

3.5 Construction of Iterative Ensembles

When constructing an ensemble of classifiers, a *chunk-based approach* was implemented as one of the training methods described in Section 3.4.1 and 3.4.2. During training on the first incoming batch of data, a single classifier was trained and used to cast predictions on the next incoming batch of data. The classifier was then added to the ensemble. For all subsequent iterations of receiving a new batch of data, the following regime was adopted:

1. A new classifier is trained on the current training set.
2. Let all classifiers (both the new classifier and the other ensemble members) make predictions on the new test set.
3. Cast the final predictions of the ensemble on instances in the test set, based on the individual predictions of the ensemble members, by using one of the following regimes:
   (a) *Majority vote*, arbitrarily choosing the class of an instance in case of a tie.
   (b) *Weighted majority vote* where each classifier votes for a class, adding a vote with a weight equal to the classifier's accuracy on the former test set.
4. Depending on whether the ensemble is full:
   (a) If the ensemble *is not* full, add the new classifier to the ensemble.
   (b) If the ensemble *is* full, exchange the new classifier with the classifier which had the lowest accuracy on the test set, given that that new classifiers accuracy was higher.

The regime is rooted in a chunked-based ensemble approach-algorithm, called *Streaming Ensemble Algorithm* (SEA) [18]. The arguments for SEA is that it is a well-known [7] approach, suited for this project because:

- Training a new classifier on the next chunk of data, evaluating and then potentially replacing one of the old classifiers, allows for updating the ensemble model for an infinite amount of time, while maintaining the ensemble size.
- Data can be discarded after fulfilling the purpose of training classifiers.

However, the regime differs from SEA in two aspects:

- Only one data set, serving as both a training and test set, is used in SEA. Instead, SEA trains a new classifier on the set and then evaluates the classifier trained during the former iteration on the set.
- SEA uses a *Quality score* instead of accuracy to evaluate the classifiers and decide when to replace a classifier in the ensemble with a new classifier. This score reflects the performance of a classifier but also takes into account the performances of other classifiers.

Regarding the first aspect: The algorithm was adjusted to incorporate both a training and a test set in order to work with the incremental training regime of this project. Instead of evaluating the former classifier on the test set, the new classifier is trained on the training set and then evaluated on the test set. These methods are hence equivalent.

Regarding the second aspect: A review of studies on iterative ensembles [7] argues that SEA is partly successful in avoiding overfitting due to the Quality Score. It favours classifiers that make an accurate prediction while others do not. Since the aim of this project is not to optimize one regime, *accuracy* instead of the *Quality Score* was used as an evaluation metric for the sake of simplicity. In addition, [7] argues that the Quality Score may make a model slow in adapting to new concepts, so not using the Quality Score will potentially make the ensemble model adapt more quickly to changes in RSC distribution.

Additionally, SEA uses a majority vote for deciding on the overall predictions of the ensemble. Since a weighted majority vote is a common alternative [7], such an approach was implemented as described above, in an attempt to further improve the models performance. Accuracy was chosen as the weight of the votes for simplicity and interpretability.

3.6 Hyperparameter Tuning

3.6.1 Dataset Partitioning

To mitigate the potential risk for bias in hyperparameters that might favour an overrepresented category, the dataset was partitioned into two subsets. The partitioning was achieved through software provided by Klimator which compared the occurrences of each RSC category within each session and combined several sessions to achieve a weighted and equal representation of the categories. The partitioning resulted in two subsets, one containing 6 sessions which were set aside for hyperparameter tuning, and one containing the remaining 14 sessions which were later used for incremental training and assessment of the models. The RSC distribution can be seen in Tables 3.2 and 3.3 for the hyperparameter set and assessment set. For specific session breakdown, see Section A.2.1, and A.2.2 in the Appendix.

RCS	**Grey Ice**	**Slush**	**Snow**	**Wet**	**Damp**	**Dry**	**Total**
Data Points	70,923	7,580	108,285	38,190	115,412	61,450	401,840
Percentage (%)	17.6%	1.9%	27.0%	9.5%	28.7%	15.3%	100.0%

Table 3.2: Distribution of data points: Hyperparameter set. Displaying the distribution of data points and percentage by RCS condition in hyperparameter set

RCS	**Grey Ice**	**Slush**	**Snow**	**Wet**	**Damp**	**Dry**	**Total**
Data Points	276,293	27,422	450,348	164,933	451,237	241,347	1,611,580
Percentage (%)	17.14%	1.70%	27.94%	10.22%	28.0%	15.0%	100.0%

Table 3.3: Distribution of data points: Assessment set. Displaying the distribution of data points and percentage by RCS condition in assessment set

3.6.2 Hyperparameter Tuning Procedure

The machine learning models evaluated in this project each come with a set of hyperparameters that configure their specific behaviour. Depending on the structure of the data, certain configuration combinations may perform better than others. To evaluate each ML model properly, tuning their respective hyperparameters was essential [44].

To tune the hyperparameters for each model, this project created its own software and implemented a grid search-based approach. The procedure of grid search includes systematically testing some or all hyperparameters of a model to identify the combination of parameter values that achieves the best performance. The metric accuracy was used to measure the performance. In the process of the hyperparameter tuning, the most interesting hyperparameters for each model were selected based on previous studies. See each model's section below for specific parameters, or Table A.1 in the Appendix. Some hyperparameters take on integer or float values this project will refer to them as discrete, while others are categorical, and constrained to a specific set of values [4]. To test potentially better-performing settings the values for the discrete hyperparameters were initially chosen from an interval centred around default values specific to the models. All available options for categorical hyperparameters were tested, except for some models. See Table A.1 in the Appendix for details.

The combinations of hyperparameters were systematically tested using grid-search and the accuracy was evaluated through predictions with the current model. The procedure for training and testing was identical to the procedures described in Section 3.4. To determine the best-performing model the software ran each hyperparameter combination on each of the 6 sessions in the hyperparameter set, evaluating the accuracy for each epoch and combining the epoch accuracy into an average accuracy for the whole session. The combination with the highest average accuracy for each session was given the highest points. With 10 hyperparameter combinations competing, the best performing during one session would be given 10 points, and the worst 1 point. In cases where multiple combinations had the same average accuracy, they were given equal points. The combination with the highest score was chosen as hyperparameter values, and in some cases entered a second tuning iteration. See Table A.1 in the Appendix for further specifications. The second tuning iteration was conducted for the models that demonstrated swift execution. During this second tuning iteration, the parameters were refined around the best-performing combination of the first iteration, and the tuning procedure was conducted again. Finally, in the case of a tie between multiple combinations, previous studies determined which one was best suited or other arguments were made for the chosen combination.

3.6.2.1 Tuning Procedure Evaluation

To evaluate this project's score-based tuning system of hyperparameters, two alternative approaches were implemented and tested on one of the models. One approach added the average accuracy for each combination and session into a total average accuracy over all sessions. The combination with the highest accuracy was declared the winner. The other approach applied a Z-score normalization to measure the deviation in each combination's accuracy. Z-score is a statistical technique that indicates the number of standard deviations a data point is from the mean. It is used for normalization purposes [45]. In this case, the data point is the average accuracy of a hyperparameter combination

for one session, denoted with a. The equation to calculate the Z-score of the average accuracy a for a combination, with the standard deviation of that session σ and the mean of all combinations during that session μ is shown below.

$$Z = \frac{a - \mu}{\sigma} \tag{3.1}$$

This project added all z values for a combination, creating a total Z-score according to the following equation. Where m is the number of sessions, and a_{ci} is the average accuracy score of one combination from the group of combinations C, for session i.

$$Z_{\text{tot}} = \sum_{i=1}^{m} \frac{a_{ci} - \mu_i}{\sigma_i}, \quad c \in C. \tag{3.2}$$

The winning combination was the one with the highest total Z-score. When comparing the different approaches this project could conclude that different winners were appointed, but that the top 3 combinations were the same. The margins between these candidates were small or insignificant, suggesting high consistency between the approaches. The evaluation of the score-based tuning procedure led to its continuous use since it demonstrates similar performance to the other approaches. The strength of this approach is aligned with the Z-score approach, in terms that it weighs the results from each session equally and does not overvalue a combination that excels in a single session, thus minimizing the risk of bias towards certain sessions.

3.6.3 Tuning Hyperparameters of an Ensemble

In case a classifier was used in an ensemble, there were ensemble-specific hyper-parameters that were optimized. These consisted of *sliding window*, *ensemble size*, *vote type* and *accuracy type*. Two options for vote type and accuracy type were investigated. Vote type consisted of either using a *weighted* or *equal* majority vote when deciding on the overall prediction of the ensemble, as described in Section 3.5. Accuracy type consisted of using either regular accuracy or a *balanced accuracy* as metric when evaluating whether to exchange a new classifier with one of the existing members of the ensemble. Regular accuracy refers to the accuracy value computed using the default formula (Formula 2.9) described in Section 2.4.1. Balanced accuracy distinguishes itself by adjusting the computation of TP, TN, FP and FN for class imbalance following a procedure described in Section 3.6.5, before using the formula (Formula 2.10) for accuracy. This was an attempt to make the model more equipped for handling concept drifts. Sliding window, on the other hand, is a boolean variable corresponding to either using a sliding window or not. This determined which of the two training routines, described in Section 3.4.1 and 3.4.2, was used. Hence, using a sliding window corresponds to an epoch time of 2 seconds and a window width of 5 epochs, whereas not using a sliding window equals an epoch time of 10 seconds. Ensemble size on the other hand, equals the number of classifiers stored in the ensemble. The size of an ensemble has been shown to impact how fast to recover from a concept drift [7], making it an important hyperparameter to tune.

3.6.4 Model Specific Hyperparameters

For each model, classifier-specific hyperparameters were tuned. In case the model constituted an ensemble of classifiers, ensemble-specific hyperparameters were optimized in conjunction. All hyperparameters specific to each model are accounted for in the following sections. For specific tuning intervals and the number of tuning iterations for each model, see Table A.1 in the Appendix.

3.6.4.1 Gaussian Naive Bayes

The hyperparameter used in the Gaussian Naive Bayes was *Smoothing*. Smoothing, also called Laplace smoothing prevents errors in the classification process if the probability value for a category becomes zero. The smoothing value is then added during the classification process to prevent zero division [46]. The variable smoothing was selected since it was the parameter of choice by Arden and Safitri in their comparison of hyperparameter tuning [47]. An additional parameter not used during tuning but specified for clarification was *Priors*. Priors set the prior probabilities for the classes before starting to train the model [8]. For clarity, this project kept the default value of 'None', indicating that all classes are assigned the same priority. The selected hyperparameters can be seen in Table 3.4.

GNB

Hyper Parameter	Selected Value	Default Value
Smoothing	0.3	10^{-9}
Priors	None	None

Table 3.4: Hyperparameters: GNB. Displaying hyperparameters used for the GNB model along with their selected and default values as specified in [8].

3.6.4.2 Complement Naive Bayes

The hyperparameters used in the Complement Naive Bayes model were *Alpha*, *Force alpha* and *Norm*. In this model, the smoothing value is referred to as the alpha value. The alpha or smoothing value serves the same purpose as in Gaussian Naive Bayes, to prevent zero division during the classification process. Force alpha is a boolean that ensures the alpha value is at least 10^{-10} or the manually given value. The parameters norm is a boolean that if set to true carries out a second normalization of the model's weights [9]. These parameters were chosen based on being selected of Dewi and Chen's when using CNB as a classifier [48]. A parameter not used for tuning but set for clarification was *Class prior*. This parameter states the prior priority for the classes. For clarity, this project retained the default value of 'None', indicating that all classes are assigned the same priority. The selected hyperparameters can be seen in Table 3.5.

CNB

Hyperparameter	Selected Value	Default Value
Alpha	1000	1.0
Force Alpha	True	True
Norm	True	False
Class prior	None	None

Table 3.5: Hyperparameters: CNB. Displaying hyperparameters used for the CNB model along with their selected and default values as specified in [9].

3.6.4.3 HoeffdingTreeClassifier

The hyperparameters used in the Hoeffding tree were *grace period*, *leaf prediction* and *split confidence*. The grace period is the number of data points that must be processed before a leaf is evaluated for a potential split. The Leaf learner is the method used within a leaf to predict the category and the split confidence is determining the acceptable level of error when deciding to initiate a split. A low value indicates low error tolerance [33]. These specific parameters were chosen based on the selection made by Khannouz and Glatard in their study focused on data stream classification [49]. An additional parameter that was not used during tuning, but specified for clarity was *Split Criterion*. The default value, info gain was kept and used for this parameter. The selected hyperparameters can be seen in Table 3.6.

HTC

Hyperparameter	Selected Value	Default Value
Grace Period	275	200
Leaf Learner	nba	nba
Split Confidence	0.3	10^{-7}
Split Criterion	info gain	info gain

Table 3.6: Hyperparameters: HTC. Displaying hyperparameters used for the HTC model along with their selected and default values as specified in [33].

3.6.4.4 Logistic Regression Ensemble

The Logistic Regression Classifier (LRC) ensemble's parameters were tuned with the ensemble as an entity, following the procedure described in Section 3.5. The hyperparamters used for tuning were *Penalty*, *C Solver* and *L1 ratio*. The three first parameters were chosen since being used by [44], and the L1 ratio was included since needed to be specified when running certain parameter combinations. It specifies the ratio between the different penalty alternatives. The parameter penalty specifies the regularization technique used for preventing overfitting [4, 12]. Solver specifies the algorithm used in the optimization problem and C specifies the regularization strength. For multiclass problems, as addressed by this project, only a subset of the solver algorithms were available. Additionally, the solver algorithm only worked with specific combinations of penalties [12]. All functional combinations were tested during tuning, see Table A.1. A hyperparameter that was not used for tuning but was changed from its default value was *Max iterations*. Max iterations specify how many iterations the solvers can take at maximum to converge. After receiving several warnings of non-converging the value was increased from 100 to 1000. Further ensemble-specific parameters were used for tuning and are described in Section 3.6.3. For clarity, this project used a penalty set to elastic-net, and the L1 ratio set to 0.5 which indicates that the penalty becomes a combination between the two available penalties, L1 (Lasso) and L2 (Ridge) [12], where both were given equal impact. Combining the advantages of the two penalties can be more effective than just using one alone [50]. The selected hyperparameters can be seen in Table 3.7.

E-LRC

Hyperparameter	Selected Value	Default Value
Penalty	Elastic-Net	L2
Solver	Saga	lbfgs
C Value	10.0	1.0
Max iterations	1000	100
L1 Ratio	0.5	None
Ensemble specific parameters		
Voting Type	Weighted	
Accuracy Type	Regular	
Ensemble Size	15	
Sliding Window	True	

Table 3.7: Hyperparameters: E-LRC. Displaying hyperparameters used for the E-LRC model along with their selected and default values as specified in [12].

3.6.4.5 Decision Tree ensemble

The Decision Tree ensemble was run as an entity to tune the parameters, following the procedure described in 3.5. The hyperparameters used in the trees were *Criterion*, *Max Depth*, *Min Samples Split*, *Min Samples Leaf* and *Max Features*. These parameters were chosen since it was used by Shami and Yang in their study of hyperparameter optimization of different ML algorithms [44]. Another parameter utilized for clarity but not used during tuning was *Class weight*, which states the weights associated with the categories. Class weight was set to 'None', which indicates that all categories had equal weight, since in a realistic scenario, it is hard to know anything about the occurrences of the RSC classes beforehand. The criterion parameter indicates which function to use when measuring the quality of a split. Max depth states the maximum depth of the tree. Min samples split states the minimum number of samples required to split an internal node, and min samples leaf states the minimum number of samples required to be at a leaf node. The max features indicate the number of features to look at when deciding on the best split [13]. Further ensemble-specific parameters were used for tuning and are described in Section 3.6.3. The selected hyperparameters can be seen in Table 3.8.

E-DT

Hyperparameter	Selected Value	Default Value
Criterion	Log Loss	Gini
Max Depth	5	None
Min Samples Split	14	2
Min Samples Leaf	10	1
Max Features	Log2	None
Class Weight	None	None
Ensemble Specific Parameters		
Voting Type	Weighted	
Accuracy Type	Balanced	
Ensemble Size	15	
Sliding Window	True	

Table 3.8: Hyperparameters: E-DT. Displaying hyperparameters used for the E-DT model along with their selected and default values as specified in [13].

3.6.4.6 K-Nearest Neighbors Ensemble

The tuning of the parameters was conducted with the ensemble as an entity, following the procedure described in 3.5. The hyperparameters used during tuning the individual models were *N-neighbours*, *Weights* and *Metric*. These parameters were singled out since considered important by Maceda and Cruz in [11]. N-neighbours define the number of 'k' nearest neighbours to consider. Weights define the weight given to the neighbour's vote, which can be uniform or dependent on the respective distance to the point being evaluated. Metric specifies which distance-calculating metric to use. Further, ensemble-specific parameters were used during tuning and are described in Section 3.6.3. The select values can be seen in Table 3.9. This project chose to limit the options of the KNN metric parameter to only Euclidian and Manhattan, as the eight potential values would incur excessive computational time. The metrics Euclidean and Manhattan were singled out since showing promising performance, having high popularity and being algorithms with low computational overhead [51, 52]. No sliding window was used since a sliding window did not increase the performance and by using no sliding window the complexity of the model is reduced.

E-KNN

Hyperparameter	Selected Value	Default Value
N-Neighbors	15	5
Weights	Distance	Uniform
Metric	Euclidean	Minkowski
Ensemble Specific Parameters		
Voting Type	Weighted	
Accuracy Type	Regular	
Ensemble Size	15	
Sliding Window	False	

Table 3.9: Hyperparameters: E-KNN. Displaying hyperparameters used for the E-KNN model along with their selected and default values as specified in [34].

3.6.4.7 Dummy Classifier

A Dummy Classifier was utilized in three different manners. First as a single classifier for creating a minimal performance baseline, second, as an ensemble candidate for classifying RSC, and last as a supporting model in the Logistic Regression ensemble. No classifier-specific hyperparameters were tuned for this model in any use case. The parameter of interest called *Strategy* is by default set to 'Prior'. This indicates that the model always predicts the category being most frequent in the prior observed data [35]. This strategy is most useful for all applied areas within this project. Ensemble-specific parameters were tuned for the Dummy ensemble candidate, as can be seen in Table 3.10. No sliding window was used since the sliding window did not increase the performance and by not using the sliding window the complexity of the model is reduced. Regarding the Single Dummy Classifier used for establishing a baseline, the ensemble-specific parameters were not tuned. The strategy was set to prior, as seen in Table 3.11.

E-Dummy

Hyperparameter	Selected Value	Default Value
Strategy	Prior	Prior
Ensemble Specific Parameters		
Voting Type	Weighted	
Accuracy Type	Balanced	
Ensemble Size	20	
Sliding Window	False	

Table 3.10: Hyperparameters: E-Dummy. Displaying hyperparameters used for the E-Dummy model along with their selected and default values as specified in [35].

S-Dummy

Hyperparameter	Selected Value	Default Value
Strategy	Prior	Prior

Table 3.11: Hyperparameters: S-Dummy. Displaying hyperparameters used for the S-Dummy model along with their selected and default values as specified in [35].

3.6.4.8 Random Forest

Random Forest was utilized for creating a top-level baseline. The hyperparameters used for the individual trees in the Random Forest are the same as those used in the Decision Tree Classifier, see Section 3.6.4.5. Further hyperparameters specific to RF utilized were *N-estimators* and *Random state*. These parameters were not used during the tuning process but were specified in advance. N-estimators specify the number of trees to be trained in the ensemble, with a default value of 100. Random state specifies a seed that controls the randomness of the samples of data points used when building trees. The default value is 'None' but the use of a fixed value creates reproducibility when building an additional model on the same data set [36]. This project used the hyperparameter values specified in Table 3.12. The parameter max features were not optimized during the tuning process but were set in advance because of certain heuristic guidelines, see Section 2.2.6.1. 'None' was used for class weight since, in a realistic scenario, it is hard to know anything about the occurrences of the RSC classes beforehand. The choice of utilizing 100 estimators was based on giving the Random Forest significantly more trees than the decision-tree ensemble created by this project while ensuring it remains computationally feasible within a reasonable time frame. The random state value of 42 was chosen as a wink to the book "The Hitchhiker's Guide to the Galaxy".

RF

Hyperparameter	Selected Value	Default Value
N-Estimators	100	100
Random State	42	None
Decision Tree Parameters		
Criterion	Log Loss	Gini
Max Depth	10	None
Min Samples Split	14	2
Min Samples Leaf	1	1
Max Features	sqrt	None
Class Weight	None	None

Table 3.12: Hyperparameters: RF. Displaying hyperparameters used for the RF model along with their selected and default values as specified in [36].

3.6.5 Evaluating the Models Performances

In order to evaluate the performances of the models, the metrics accuracy, precision, recall and F1-score were computed utilizing the built-in functions from the scikit learn metric library [31]. Accuracy was computed according to the Formula 2.9 in Section 2.4.1. The other metrics were configured to compute a weighted average across all the categories of interest, to account for class imbalance. The weighted average is calculated by computing metrics for each RSC class individually, and weighting them based on their support, which is the number of true instances (FP+FN) for each RSC class [31]. Additionally, the total number of TP, FP, TN and FN were summarized for each epoch of a session during the training procedure of each model. These metrics were then used to compute an evaluation metric referred to as *balanced accuracy*, following a procedure described in 3.6.5.3 resulting in an accuracy adjusted for class imbalance, where all classes are given equal weight. This metric was solely used as a hyperparameter to serve as an evaluation metric when deciding on switching classifiers in the ensemble, see Section 3.6.3, in an attempt to make the models more equipped to handle concept drifts. All metrics were saved to Excel files, one such file per model and session.

3.6.5.1 Finding Strengths and Weaknesses of the Models

The average of all evaluation metrics was computed for each model and epoch to serve as a basis for comparing the performances of the models, in order to find strengths and weaknesses of the models. The average for each evaluation metric and model over each session was also computed. For further investigations, accuracy was chosen as the main metric. The main argument for using accuracy is that if a model were to be put into use in a vehicle, such a model should be as precise as possible. That is why accuracy is a relevant metric, as it accounts for the proportion of correct predictions. For each model, the mean accuracy, the median accuracy, and the 25th and 75th percentile were calculated, along with the maximum and minimum accuracy.

In order to further evaluate the models' performances, the results of each model were compared with those of the two baseline models (Single Dummy Classifier, Random Forest). Additionally, two different scenarios were explored: comparing the models' performances in handling concept drifts (see Section 3.6.5.2), and further investigating a period where all classes were present, analyzing the models' performances during such a period. Model comparisons were also performed using two graphs, where accuracies were plotted along with the data distribution. These two graphs concern two sessions: one where one class constitutes the majority for most of the session, and one with several periods of concept drifts. The main goal was to compare the models' performances on more stable data versus data with a higher variance in data distribution (indicated by concept drifts). The data distribution was visualized for each session by illustrating the percentages of each category for every epoch of a session.

3.6.5.2 Evaluating Model Performance on Concept Drifts

For each session, a plot of the data distribution was examined in order to determine epochs of concept drifts. The time for a concept drift, referred to as an *concept drift-event*, was determined visually by identifying a sudden shift in data distribution signified by the following aspects:

- From one epoch to another, there is a notable shift in which class constitutes the majority
- The class that becomes the majority has been a minority class for at least the five preceding epochs

A sufficient number of concept drift events were identified to form a diverse set, ensuring that each class became the majority on at least one occasion, during at least one concept drift event. Some epochs fulfilling the definition above were actively not labelled as concept drift-events. This was due to, for example, that the class that became the majority had been present in a significant proportion during many proceeding epochs, why it should not count as a concept drift-event. Moreover, based on the definition above, it became apparent that the models were not consistently most affected precisely at the time of the concept drift-event. Rather, they frequently had a local minimum in accuracy some epochs before or after the event. In Figures A.2 - A.15 in the Appendix, the session data distribution during the epochs is displayed along with the concept drift-events. Figure 3.6 illustrates one such event, displaying the data distribution during epochs 0 - 9 of session 3. It can be seen that a shift in data distribution often starts one or several epochs before the actual event occurs, and the class that becomes the majority often reaches a maximum one or several epochs after the event. Therefore, the performance of each model was evaluated during a *concept drift-window*. The width of such a window was set to include the epoch where the class that becomes the majority starts to increase in proportion. The window extends until such a class approaches its peak distribution. In case several concept drift-windows were connected, they were merged into one as a preceding window is likely to influence the performance of the models during a subsequent window. The concept drift-windows is also illustrated in Figures A.2 - A.15 in the Appendix. The performance of each model in terms of handling a concept drift was evaluated as follows:

1. The local minimum in accuracy during each concept drift-window
2. The average accuracy during each concept drift-window

These accuracies (local minimum and average) were evaluated by comparison between models as well as with the two baseline models (Single Dummy Classifier, Random Forest).

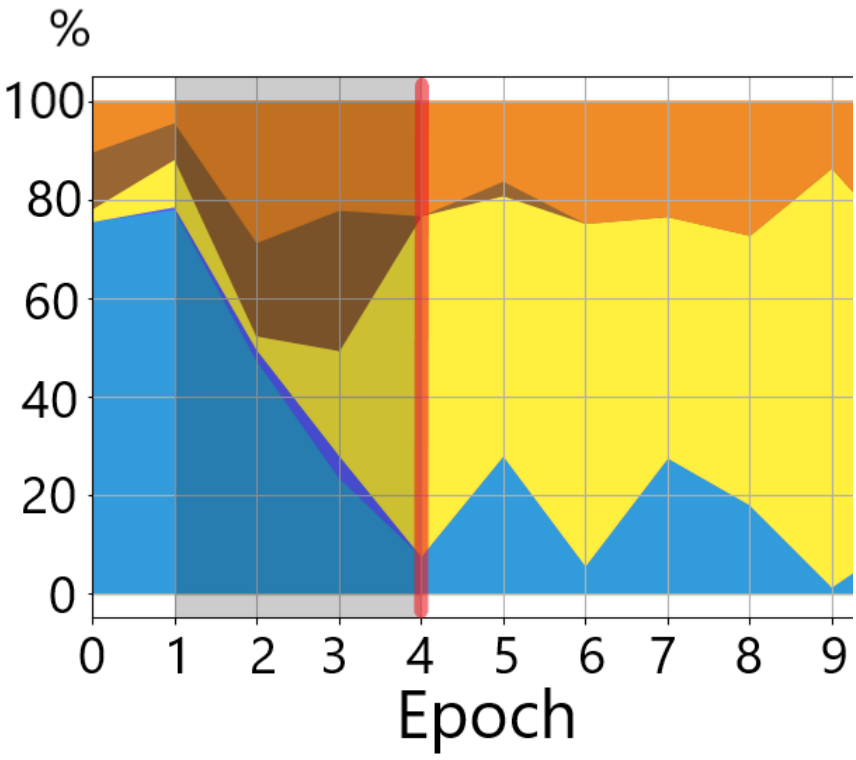


Figure 3.6: Concept drift-event. Displaying a concept drift-event in session 3, during epoch 4 with a concept drift-window from epoch 1 - 4. There is a notable shift in RSC distribution, from *damp* (light blue) to *snow* (yellow), where the proportion constituting snow is increasing during the epoch preceding the event.

3.6.5.3 Balanced Accuracy

In an attempt to make the ensemble models more equipped to handle concept drifts, a metric referred to as *balanced accuracy* was developed as an alternative when deciding on switching classifiers in an ensemble (as described in Section 3.6.3). The need for such a metric lies in that the traditional accuracy metric is biased toward majority classes due to its formula (see Formula 2.9, Section 2.4.1). Balanced accuracy was therefore formulated to serve as an accuracy measurement adjusted for class imbalance, which could improve the ensemble models' ability to adapt to shifts in data patterns, by creating a more diverse ensemble of classifiers. Subsequently, evaluating balanced accuracy when deciding on switching classifiers could potentially improve an ensemble model's performance if the data distribution is unstable.

Balanced accuracy was defined without changing the Formula (2.10) for computing the accuracy and instead changing how to compute TP, TN, FP and FN before applying the formula. These parameters are originally computed by summarizing them for each of the classes. Hence, a class that occurs more frequently makes a larger contribution compared to a less frequently occurring class. In order to let each class contribute to these parameters without depending on the prevalence of the class, the parameters were adjusted as follows:

1. TP_{Tot_B} and FN_{Tot_B} are the total number of true positives and false negatives given a balanced computation. These were computed by dividing the number of true positives and false negatives for each class by the ratio representing the class instances' share of the total number of instances. The resulting values were then summarized.

2. TN_{Tot_B} and FP_{Tot_B} are the total number of true negatives and false positives given a balanced computation. Since TN and FP truly belong to one of the other classes than the class under consideration, the computations were performed by dividing the count of true negatives and false positives for each class by the proportion of instances belonging to all other classes out of the total number of instances. Finally, summarizing the resulting values for TN_{Tot_B} and FP_{Tot_B} respectively.

Subsequently, the following formulas were formulated and used for computing TP_{Tot_B}, FP_{Tot_B}, TN_{Tot_B} and FN_{Tot_B}:

$$TP_{Tot_B} = \sum_{i=1}^{K} \frac{TP_{C_i}}{p}, \quad p = \frac{N_{C_i}}{N_{Tot}}$$

$$FP_{Tot_B} = \sum_{i=1}^{K} \frac{FP_{C_i}}{p}, \quad p = \frac{N_{Tot} - N_{C_i}}{N_{Tot}}$$

$$TN_{Tot_B} = \sum_{i=1}^{K} \frac{TN_{C_i}}{p}, \quad p = \frac{N_{Tot} - N_{C_i}}{N_{Tot}}$$

$$FN_{Tot_B} = \sum_{i=1}^{K} \frac{FN_{C_i}}{p}, \quad p = \frac{N_{C_i}}{N_{Tot}}$$

K represents the number of classes, i is the index of a class and C_i the class of index i. p is a ratio and N a number of instances. Hence, N_{C_i} is the number of instances belonging to the class with index i and N_{Tot} the total number of instances.

Using the formulas above to compute the total count of TP, FP, TN and FN ensured that the influence of each class on these parameters was independent of its size, thus providing each class with an equal opportunity to contribute proportionally. The Formula for accuracy (2.10) stated in Section 2.4.1, was then applied in order to gain the final balanced accuracy result.

4
Results

The procedures outlined in Chapter 3 aim to produce results that address the project's goals (see Section 1.3). These results are presented in this chapter, where the models' performances are evaluated by comparing their results to identify strengths and weaknesses. The findings are presented in a logical order, beginning with an overview of the models' overall performances using metrics accounted for in Section 3.6.5, followed by analyzing their performances on certain parts of the data in order to understand how the models handle different situations. The findings are then analyzed in the upcoming chapter (Chapter 5), in order to identify areas for future research and development - the third goal of the project.

4.1 Interpreting the Tables

Results are mainly presented through tables, some standardized in design to enhance interpretability. Tables applying the same standardized design require the same explanations, which is why this section conveys such details. Tables that are not explained here contain all the necessary information in their captions.

4.1.1 Tables Storing Metrics

One group of tables with a standardized design is tables storing metrics. Such tables include Table **4.1, 4.2, 4.3, 4.4, 4.9, 4.10, 4.14** and highlight models with the best performance, as well as those whose performance is in close proximity to that of the best model. Additionally, they indicate whether the models outperform the Random Forest Model and/or underperform compared to the Single Dummy Classifier. More specifically, these tables require the following explanation: The light purple background of a cell indicates the highest value among the evaluated models in each session, while light blue indicates a value that is within 1 % of the highest value for that session. The green up-arrow (∧), suggests a value exceeding the high-level baseline produced by RF, and a red down-arrow (v) indicates a lower value than the low-level baseline created by the Single Dummy Classifier (S-Dummy), during that session. In Table 5.1 the orange colour indicates having the lowest metric value decrease, suggesting robustness. The principles apply to both the session values and total values.

4.1.1.1 Concept Drift: Result Tables

In addition, Table **4.9** (Concept drift: Local minimum accuracy) and **4.10** (Concept drift: Average accuracy) require further explanation. Each concept drift-window is defined in the most left column of the table. They are written on the form "Epoch *number-number* (*number/-s*)". "*Number-number*" displays the width of the window, in other words, which

epoch is the start of the window and which signifies the end of the window. "(*number/-s*)" indicates during which epoch/epochs of the concept drift-window that actual concept drift-events occur, given the definition described in Section 3.6.5.2.

4.1.2 Tables Displaying Summary of Results

There is another group of tables with a mutual design. These tables include Table **4.5**, **4.6**, **4.7**, **4.8**, **4.11** and **4.12**, and display a summary of results stored in another table (Table 4.1, 4.2, 4.3, 4.4, 4.9 and 4.10). Each table cell displays both a percentage as well as a proportion written inside brackets. For example in Table 4.11, each percentage and proportion correspond to the proportion of concept drift-windows that fulfill the statement stated in the left column, for each of the models. Hence, "*Best performer*" is the proportion of concept drift-windows in which a model outperforms the other models in terms of accuracy (proportion of purple cells). "*Best performer & in close proximity (<= 1%) to the best*" on the other hand, also includes the proportion of accuracies within 1% of the model having the highest accuracy during a concept drift-window (proportion of both purple and blue cells). In the same manner "*Outperforming RF* ($\wedge$)" and "*Outperformed by S-Dummy* (v)" correspond to the proportion of concept drift-windows that have a higher accuracy than the Random Forest model (proportion of green up-arrows ($\wedge$)) and a lower accuracy than the Single Dummy Classifier (proportion of red down-arrows (v)). For each of these statements, the cell of the model with the highest percentage is highlighted: A purple colour indicates the best performance whereas a red colour indicates the worst performance.

4.2 Analysis of Average Metrics

To estimate the strengths and weaknesses of the models, comparing the overall performance between models is of value. The overall performance can be measured by the average of metrics over all sessions among all the models. The following tables present such average metrics: Table 4.1 (average accuracy), Table 4.2 (average recall), Table 4.3 (average precision), and Table 4.4 (average F1-score). All models were run with the best-performing hyperparameter combination as seen in Table A.1, and were run according to the training procedure described in Section 3.4. This means that E-LRC and E-DT used a sliding window and a 2-second epoch time.

The first notable result is the demonstrated strength from E-LRC and E-DT regarding average accuracy. Since accuracy and weighted recall are identical according to [31], then what is said about average accuracy, also applies to average recall. As seen in the summary Table 4.5 for accuracy, and 4.6 for recall, E-LRC has the strongest performance in 9 of the 14 sessions and is the best or in close proximity in 13 of them. E-DT has the strongest performance in the remaining sessions and is the best or in close proximity in all of them. E-LRC and E-DT both outperform RF in nearly all sessions. They are both outperformed by S-Dummy once in session 14. Notable performances are HTC and E-KNN, which outperform RF 6 times each. Finally, E-Dummy outperforms RF 4 times, and GNB outperforms RF 3 times. CNB and E-Dummy show relatively weak performances and are outperformed by S-Dummy in 11 and 12 of the sessions, respectively. Regarding the total average accuracy over all sessions, as seen in table 4.1 and 4.2, the strongest performing model is E-DT with a total average of 68.82 %. E-LRC is close behind with

68.81 %.

HTC is the notable top performer when considering average precision. Based on the summary Table 4.7 for precision, HTC performs best in half of the sessions and outperforms RF in 50.00 %. GNB, CNB and E-DT all perform best in 2 sessions each, and E-RLC performs best during one session. When taking into account being best or in close proximity of the best, HTC shows strength in 8 sessions, followed by E-LRC with 6 sessions, GNB with 5 sessions, E-DT with 4 sessions and E-KNN with one session. The highest total average precision as seen in Table 4.3 is held by HTC with 69.24 %.

Regarding the average F1-score, summarized in Table 4.8, E-LRC and E-DT show strong performance with win rates of 7 and 4 out of 14, respectively. They are both best or in close proximity to being best in 12 of the sessions. E-LRC outperforms RF in 12 of the sessions, and E-DT outperforms RF in 11 sessions. Notable is that HTC also outperforms RF in 12 of the sessions, and wins 2 sessions. HTC is also the best or in close proximity to being the best in half of the sessions. The last win goes to E-KNN, which performs best during session 5. E-Dummy shows a weak performance and is outperformed by S-Dummy in 10 of the sessions. Regarding the total average F1-score, the best performer is E-LRC with 64.61 %, closely followed by E-DT with 64.45 %. HTC is also showing some strength with a total average of 63.14 %, as seen in Table 4.4.

Avg Accuracy	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN	S-Dummy	RF
Session 1	71.25%	70.54%	71.15%	68.27% v	73.99% ^	73.88% ^	69.88%	68.79%	71.59%
Session 2	80.71%	77.38% v	82.88% ^	72.22% v	84.89% ^	84.30% ^	77.65% v	77.85%	81.03%
Session 3	53.20%	46.10% v	52.94%	44.56% v	56.06% ^	55.78% ^	48.93% v	49.55%	54.03%
Session 4	57.82%	48.69% v	60.98% ^	51.79%	63.24% ^	63.29% ^	54.91%	51.55%	58.07%
Session 5	58.05%	49.42% v	57.53%	49.29% v	58.74%	60.21% ^	58.86%	52.27%	59.57%
Session 6	58.47%	59.00%	59.42%	58.77%	62.32% ^	61.98% ^	60.79% ^	56.75%	60.06%
Session 7	56.70% v	52.30% v	62.44% ^ v	64.68% ^ v	70.73% ^	70.99% ^	64.08% ^ v	66.91% ^	58.88% v
Session 8	62.98%	45.33% v	64.52%	56.59% v	68.25% ^	68.24% ^	60.51%	56.67%	64.52%
Session 9	53.00% v	39.57% v	58.94% ^	56.85% v	66.83% ^	67.57% ^	59.66% ^	58.83% ^	58.57% v
Session 10	67.08% ^	63.47%	64.97%	59.54% v	68.79% ^	68.07% ^	64.98%	62.84%	66.33%
Session 11	72.99% ^ v	66.31% v	71.48% v	70.71% v	74.49% ^	74.26% ^	70.43% v	73.79% ^	72.59% v
Session 12	67.80% ^ v	63.53% v	72.86% ^ v	79.32% ^ v	82.18% ^	82.24% ^	78.95% ^ v	81.92% ^	67.53% v
Session 13	57.86% v	58.63% v	59.02% v	60.77% ^ v	64.51% ^	64.48% ^	61.25% ^ v	61.90% ^	60.02% v
Session 14	40.73% v	37.50% v	56.52% ^ v	60.89% ^ v	68.32% ^ v	68.21% ^ v	56.55% ^ v	69.79% ^	46.91% v
Total	61.33% v	55.56% v	63.98%^	61.02% v	68.81% ^	68.82% ^	63.39% ^ v	63.53% ^	62.84% v

Table 4.1: Average Accuracy among all models, session by session. Displaying the average accuracy calculated for each model, session by session, together with the total average accuracy over all sessions. Following the explanation given in Section 4.1.1.

Avg Recall	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN	S-Dummy	RF
Session 1	71.25 %	70.54%	71.15%	68.27% v	73.99% ^	73.88% ^	69.88%	68.79%	71.59%
Session 2	80.71%	77.38% v	82.88% ^	72.22% v	84.89% ^	84.30% ^	77.65% v	77.85%	81.03 %
Session 3	53.20%	46.10% v	52.94%	44.56% v	56.06% ^	55.78% ^	48.93% v	49.55%	54.03 %
Session 4	57.82%	48.69% v	60.98% ^	51.79%	63.24% ^	63.29% ^	54.91%	51.55%	58.07%
Session 5	58.05%	49.42% v	57.53%	49.29% v	58.74%	60.21% ^	58.86%	52.27%	59.57%
Session 6	58.47%	59.00%	59.42%	58.77%	62.32% ^	61.98% ^	60.79% ^	56.75%	60.06%
Session 7	56.70% v	52.30% v	62.44% ^ v	64.68% ^ v	70.73% ^	70.99% ^	64.08% ^ v	66.91% ^	58.88% v
Session 8	62.98%	45.33% v	64.52%	56.59% v	68.25% ^	68.24% ^	60.51%	56.67%	64.52%
Session 9	53.00% v	39.57% v	58.94 % ^	56.85% v	66.83% ^	67.57% ^	59.66% ^	58.83% ^	58.57% v
Session 10	67.08% ^	63.47%	64.97%	59.54% v	68.79% ^	68.07% ^	64.98%	62.84%	66.33%
Session 11	72.99% ^ v	66.31% v	71.48% v	70.71% v	74.49% ^	74.26% ^	70.43% v	73.79% ^	72.59% v
Session 12	67.80% ^ v	63.53% v	72.86% ^ v	79.32% ^ v	82.18% ^	82.24% ^	78.95% ^ v	81.92% ^	67.53% v
Session 13	57.86% v	58.63% v	59.02% v	60.77% ^ v	64.51% ^	64.48% ^	61.25% ^ v	61.90% ^	60.02% v
Session 14	40.73% v	37.50% v	56.52% ^ v	60.89% ^ v	68.32% ^ v	68.21% ^ v	56.55% ^ v	69.79% ^	46.91% v
Total	61.33% v	55.56% v	63.98% ^	61.02% v	68.81% ^	68.82% ^	63.39% ^ v	63.53% ^	62.84% v

Table 4.2: Average Recall among all models, session by session. Displaying the average recall calculated for each model, session by session, together with the total average recall over all sessions. Following the explanation given in Section 4.1.1.

Avg Precision	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN	S-Dummy	RF
Session 1	68.33%	66.05%	71.11%	56.28%	70.45%	70.91%	67.41%	56.01%	71.39%
Session 2	87.11% ^	83.47%	85.61%	67.02% v	84.66%	83.66%	75.54%	72.00%	87.10%
Session 3	47.83%	43.74%	55.84% ^	25.61% v	54.65% ^	53.08%	42.83%	30.09%	53.74%
Session 4	63.97%	62.61%	65.36%	40.56%	64.90%	64.17%	59.23%	39.63%	65.73%
Session 5	61.03%	59.27%	59.98%	26.78% v	61.00%	61.34%	59.71%	29.89%	61.63%
Session 6	60.81%	61.39%	63.29% ^	37.82%	60.29%	60.26%	61.31%	35.86%	62.55%
Session 7	64.25%	65.38%	69.52% ^	51.34% v	71.46% ^	71.33% ^	63.27%	52.32%	69.48%
Session 8	67.94% ^	62.19%	68.34% ^	42.75%	67.29%	66.12%	58.78%	42.49%	67.79%
Session 9	68.50% ^	59.82%	66.69 %	40.17% v	67.59%	66.45%	62.61%	41.85%	67.79%
Session 10	69.20% ^	66.84%	68.13%	44.81% v	69.10%	69.45% ^	68.46%	47.35%	69.12%
Session 11	71.93%	73.63%	73.55%	57.01% v	67.05%	67.16%	61.80%	60.15%	73.70%
Session 12	74.87%	74.95%	79.83% ^	70.22% v	77.96%	77.64%	73.23%	72.45%	79.41%
Session 13	63.40%	68.51% ^	67.01% ^	45.28% v	65.48%	64.53%	60.53%	46.18%	66.72%
Session 14	72.52%	45.28% v	75.15% ^	50.86% v	68.18%	70.16%	59.85%	58.70%	74.53%
Total	67.26%	63.80%	69.24%	46.89% v	67.86%	67.59%	62.47%	48.93%	69.33%

Table 4.3: Average Precision among all models, session by session. Displaying the average precision calculated for each model, session by session, together with the total average precision over all sessions. Following the explanation given in Section 4.1.1.

Avg F1-score	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN	S-Dummy	RF
Session 1	67.60%	63.81%	68.78% ^	60.52%	69.83% ^	69.88% ^	65.81%	60.38%	68.68%
Session 2	81.53% ^	75.32%	82.74% ^	68.84% v	83.04% ^	81.90% ^	75.74%	73.92%	81.09%
Session 3	46.18%	40.56%	49.64% ^	31.57% v	49.92% ^	49.37% ^	40.19%	36.60%	47.80%
Session 4	57.74%	45.32%	60.09% ^	43.95%	60.25% ^	60.10% ^	50.84%	43.31%	58.27%
Session 5	54.29%	47.98%	55.63%	34.19% v	54.32%	55.17%	55.85%	37.42%	57.06%
Session 6	54.33%	54.41%	57.67% ^	45.25%	57.27% ^	56.71%	54.79%	43.18%	57.26%
Session 7	55.26% v	50.84% v	61.96% ^	56.28% v	67.02% ^	66.96% ^	59.36% ^	57.68%	58.43%
Session 8	61.01%	40.80% v	63.02% ^	47.15%	64.14% ^	63.59% ^	54.34%	47.05%	62.33%
Session 9	50.82%	31.63% v	58.61% ^	45.91% v	62.29% ^	62.70% ^	53.80%	47.76%	57.67%
Session 10	65.24% ^	61.96%	63.73%	49.39% v	65.58% ^	64.79% ^	60.57%	52.35%	64.27%
Session 11	68.21%	66.07%	70.48% ^	62.05% v	68.33%	68.00%	63.08% v	65.19%	68.79%
Session 12	67.31% v	63.66% v	73.60% ^ v	73.57% ^ v	78.61% ^	78.51% ^	73.48% ^ v	75.97% ^	68.61% v
Session 13	52.76%	59.26% ^	58.58% ^	50.34% v	59.16% ^	59.36% ^	53.73%	51.39%	56.24%
Session 14	40.23% v	34.64% v	59.43% ^ v	54.12% ^ v	64.77% ^	65.31% ^	52.18% ^ v	62.45% ^	50.84%
Total	58.75%	52.59% v	63.14% ^	51.65% v	64.61% ^	64.45% ^	58.13%	53.90%	61.24%

Table 4.4: Average F1-score among all models, session by session. Displaying the average F1-score calculated for each model, session by session, together with the total average F1-score over all sessions. Following the explanation given in Section 4.1.1.

Summary Avg Accuracy	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN
Best performer	**0.00%** (0/14)	**0.00%** (0/14)	**0.00%** (0/14)	**0.00%** (0/14)	**64.29%** (9/14)	**35.71%** (5/14)	**0.00%** (0/14)
Best performer AND in close proximity (<= 1%) to the best	**0.00%** (0/14)	**0.00%** (0/14)	**0.00%** (0/14)	**0.00%** (0/14)	**92.86%** (13/14)	**100.00%** (14/14)	**0.00%** (0/14)
Outperforming RF (^)	**21.43%** (3/14)	**0.00%** (0/14)	**42.86%** (6/14)	**28.57%** (4/14)	**92.86%** (13/14)	**100.00%** (14/14)	**42.86%** (6/14)
Outperformed by S-Dummy (v)	**42.86%** (6/14)	**78.57%** (11/14)	**35.71%** (5/14)	**85.71%** (12/14)	**7.14%** (1/14)	**7.14%** (1/14)	**50.00%** (7/14)

Table 4.5: Summary of statistics concerning the average Accuracy. Displaying the summarized statistics based on results stored in Table 4.4. See Section 4.1.2 for descriptions of how to interpret the table.

Summary Avg Recall	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN
Best performer	**0.00%** (0/14)	**0.00%** (0/14)	**0.00%** (0/14)	**0.00%** (0/14)	**64.29%** (9/14)	**35.71%** (5/14)	**0.00%** (0/14)
Best performer AND in close proximity (<= 1%) to the best	**0.00%** (0/14)	**0.00%** (0/14)	**0.00%** (0/14)	**0.00%** (0/14)	**92.86%** (13/14)	**100.00%** (14/14)	**0.00%** (0/14)
Outperforming RF (^)	**21.43%** (3/14)	**0.00%** (0/14)	**42.86%** (6/14)	**28.57%** (4/14)	**92.86%** (13/14)	**100.00%** (14/14)	**42.86%** (6/14)
Outperformed by S-Dummy (v)	**42.86%** (6/14)	**78.57%** (11/14)	**35.71%** (5/14)	**85.71%** (12/14)	**7.14%** (1/14)	**7.14%** (1/14)	**50.00%** (7/14)

Table 4.6: Summary of statistics concerning the average Recall. Displaying the summarized statistics based on results stored in Table 4.2. See Section 4.1.2 for descriptions of how to interpret the table.

Summary Avg Precision	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN
Best performer	14.29% (2/14)	14.29% (2/14)	50.00% (7/14)	0.00% (0/14)	7.14% (1/14)	14.29% (2/14)	0.00% (0/14)
Best performer AND in close proximity (<= 1%) to the best	35.71% (5/14)	14.29% (2/14)	57.14% (8/14)	0.00% (0/14)	42.86% (6/14)	28.57% (4/14)	7.14% (1/14)
Outperforming RF (^)	28.57% (4/14)	7.14% (1/14)	50.00% (7/14)	0.00% (0/14)	14.29% (2/14)	14.29% (2/14)	0.00% (0/14)
Outperformed by S-Dummy (v)	0.00% (0/14)	7.14% (1/14)	0.00% (0/14)	71.43% (10/14)	0.00% (0/14)	0.00% (0/14)	0.00% (0/14)

Table 4.7: Summary of statistics concerning the average Precision. Displaying the summarized statistics based on results stored in Table 4.3. See Section 4.1.2 for descriptions of how to interpret the table.

Summary Avg F1-score	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN
Best performer	0.00% (0/14)	0.00% (0/14)	14.29% (2/14)	0.00% (0/14)	50.00% (7/14)	28.57% (4/14)	7.14% (1/14)
Best performer AND in close proximity (<= 1%) to the best	7.14% (1/14)	7.14% (1/14)	50.00% (7/14)	0.00% (0/14)	85.71% (12/14)	85.71% (12/14)	7.14% (1/14)
Outperforming RF (^)	14.29% (2/14)	7.14% (1/14)	85.71% (12/14)	14.29% (2/14)	85.71% (12/14)	78.57% (11/14)	21.43% (3/14)
Outperformed by S-Dummy (v)	21.43% (3/14)	35.71% (5/14)	14.29% (2/14)	71.43% (10/14)	0.00% (0/14)	0.00% (0/14)	21.43% (3/14)

Table 4.8: Summary of statistics concerning the average F1-score. Displaying the summarized statistics based on results stored in Table 4.4. See Section 4.1.2 for descriptions of how to interpret the table.

4.3 Boxplot of the Models Accuracy Over Sessions

Since average metrics do not convey how the performance of a model varies, an important aspect to evaluate is the range that each model exhibits across different sessions and epochs. For this purpose, Figure 4.1 shows a boxplot of the accuracy over all epochs in all sessions among all the models. The accuracy values come from the models running with their best-performing hyperparameter combination as seen in Table A.1. The top line indicates the 100th percentile and the bottom line represents the minimum value (0 percentile). The bottom of the box corresponds to the 25th percentile, and the top to the 75th. The middle line denotes the median, with the corresponding value to the left. The x denotes the location for the average, starting with μ. If the average value is presented below the median line, it is lower than the median, and if presented above the line, it is higher.

A notable finding is that the 75th percentile and the 50th percentile of E-LRC and E-DTs are significantly higher than the rest of the models, including the baseline models. E-LRC has a 75th percentile at 97.32 %, which is 5.46% above that of the third-ranked S-Dummy, whereas E-DT is 5.47% above the S-Dummy. Regarding the 50th percentile, E-LRC is 6.14 and E-DT 6.2% above the third-ranked S-Dummy. Their 25th percentile is also relatively high but is approximately in line with HTC's and RF's 25th percentile. Notable is also HTC and RF being the only models never reaching a minimum accuracy of 0.00 % but staying at 7.96% and 3.12%, respectively. All models reach at some points during the sessions an accuracy of 100.00%. GNB, HTC, and RF show a shorter box, indicating less variability among the accuracy values, suggesting stability. RF wins with the smallest span of 33.91%, closely followed by HTC with 34.02 and GNB with 36.05. While, the other models show larger boxes, indicating a higher variation among the values. E-Dummy shows the largest variety of all the models with a span of 51.98 %, while CNB holds the lowest 25th percentile with 34.49 %, indicating weaker performances.

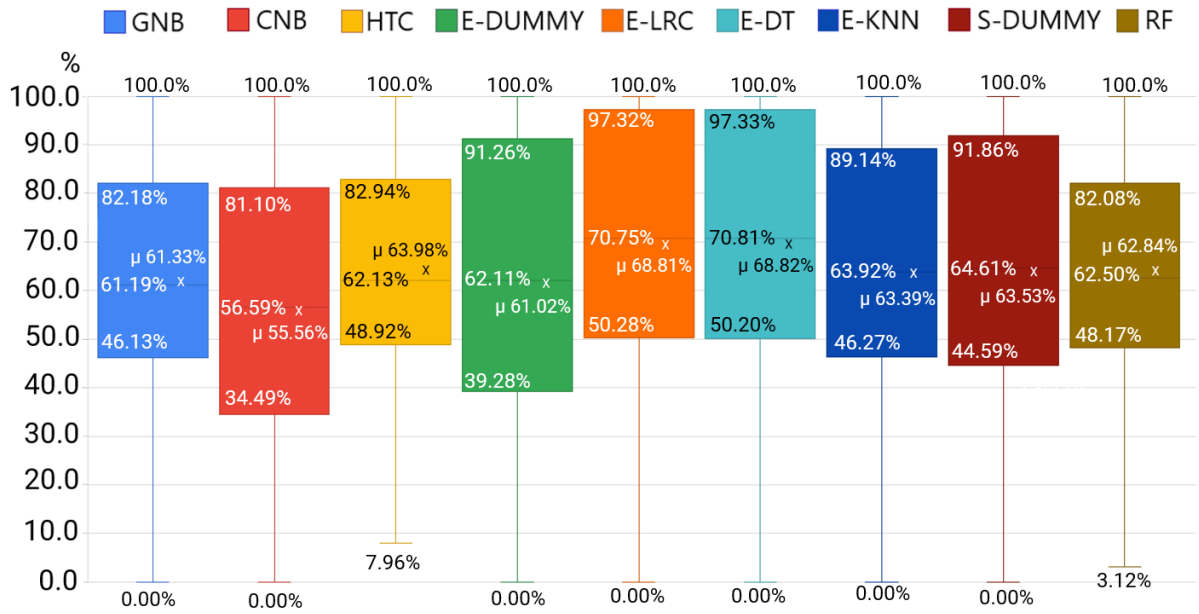


Figure 4.1: Boxplot: Average accuracy. Boxplot displaying the distribution of each model's accuracy for every epoch in all fourteen evaluated sessions. Showing the 75th, 50th and 25th percentile values to the left inside the box, along with the total average beginning with μ. The 100th and 0th percentile values are indicated at the beginning of each boxplot line

4.4 Accuracy Visualizations for Sessions 2 and 4

In an attempt to evaluate the reasons behind the fluctuations in accuracy across all models, and to further identify the situations in which each model demonstrates strength or weakness, two sessions were investigated more thoroughly. Figures 4.3 and 4.4 show all the evaluated models' accuracy, epoch by epoch, during sessions 2 and 4. The RSC distribution is visible in the background, with the colour coding explained by Figure 4.2 below. Epoch 0, the first epoch, serves as a training set and is not evaluated; therefore, the curves begin at epoch 1.

Session 2 is a session with relatively low variation in RSC, meaning that the condition is not often changed and that a condition tends to dominate most of the time. During 72.9 % of the time, one of the occurring RSCs dry, snow or grey ice dominates. In this session, the following epochs, pointed out by a vertical red line, E9, E21, E42, E52 and E56, all indicate a concept drift event. The grey area around these epochs corresponds to a concept drift-window. Sessions 2's data distribution can be seen in Appendix A.3. The general analysis of concept drift is seen in Section 4.5

Session 4 has a relatively high variation in RSC, with 10 marked concept drift-events, at epochs 2, 3, 12, 15, 20, 22, 24, 34, 42, and 43. It also contains a time frame with all six RSCs present simultaneously during epochs 5 to 8, which is further analysed in Section 4.6. Sessions 4's data distribution can be seen in Appendix A.5.

In both figures, it is notable that when one RSC is close to 100.00 %, which can be seen in session 2, epochs 11 to 17, most models achieve an accuracy close to 100.00 %. However, during concept drift-events, the accuracy decreases for all models, though some models show a smaller decline. It is also notable, that during time frames with no dominating RSC, but rather with a high variety of different RSC, such as session 4, epochs 5 to 8, and session 2, epochs 52 to 59, all models struggle, with some struggling more than others. A notable weak performer during such time frames is CNB, and a relatively consistent and strong performance is demonstrated by HTC.

Figure 4.2: RSC legend. Presenting the colour coding for each RSC and the condition codes used in the data set.

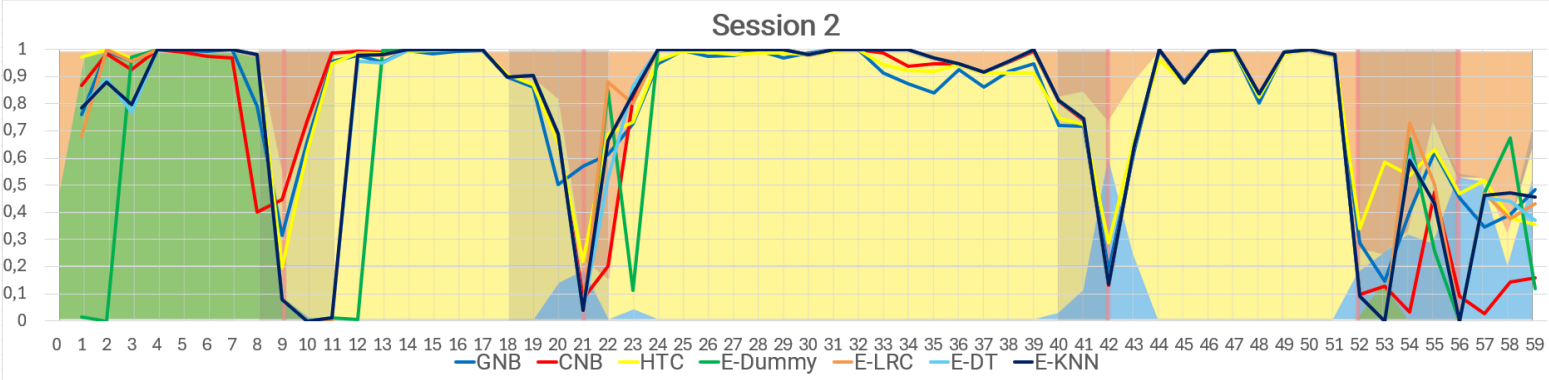


Figure 4.3: Visualization of Accuracy: Session 2. Displaying the accuracy epoch by epoch for all models with RSC distribution of session 2 in the foreground. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

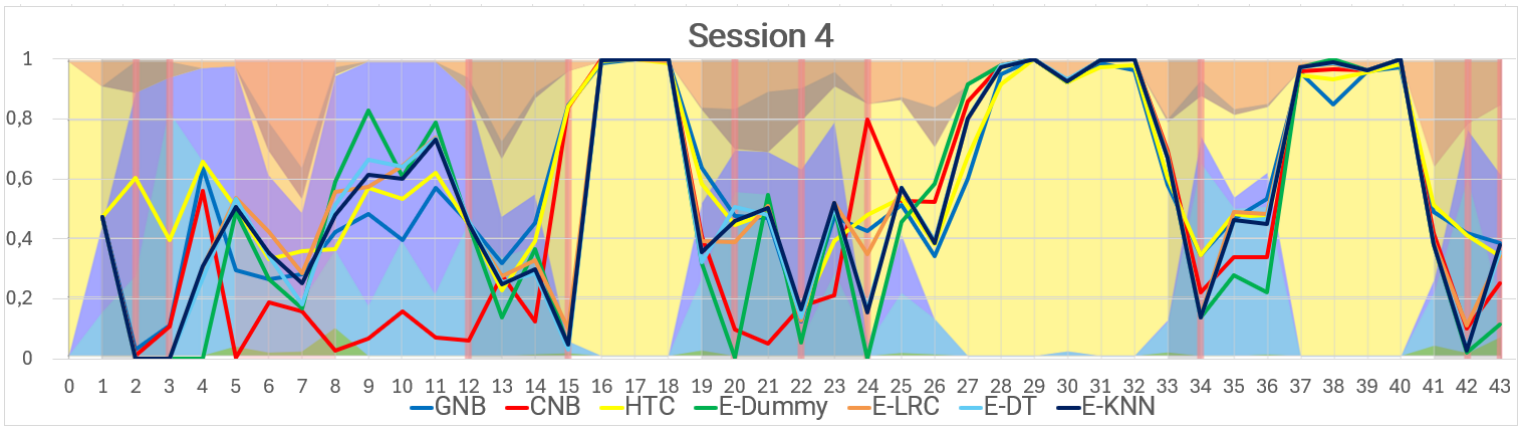


Figure 4.4: Visualization of Accuracy: Session 4. Displaying the accuracy epoch by epoch for all models with RSC distribution of session 4 in the foreground. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

4.5 Handling Concept Drifts

As seen in the former section (Section 4.4) the models all struggle to maintain accuracy when there is a shift in RSC. Therefore a potential strength to evaluate is how the models handle concept drifts. This was examined according to the procedure described in Section 3.6.5.2, by identifying concept drift-events and the concept drift-windows surrounding the events. Average accuracies as well as local minimum accuracies during such windows are displayed in Table 4.9 (Concept drift: Local minimum accuracy) and 4.10 (Concept drift: Average accuracy) for each of the models. The results are summarized in Table 4.11 and 4.12. The effect on model accuracies during concept drifts are visualized in two graphs, Figure 4.3 (session 2) and Figure 4.4 (session 4), in Section 4.4.

The overall best performer was HTC followed by GNB. This notion is based on these models having the highest and second-highest proportions of achieving the highest average and highest local minimum accuracy during the concept drift-windows, or being in close proximity to those averages, as well as outperforming RF. Having the highest proportion of such achievements are referred to as "wins". The "worst performer", on the other hand, is CNB, given that it has been outperformed by the S-Dummy most times, both in terms of local minimum accuracy and average accuracy over the concept drift-windows. Likewise, CNB has the lowest average "average accuracy" over all concept drift-windows. Although E-DT and E-KNN have a slightly higher average "average accuracy", they share the position with CNB of being outperformed by the S-Dummy the most times in terms of average accuracy during concept drift-windows. Both CNB and the ensembles in general had low proportions of wins and high proportions of being outperformed by the S-Dummy. In average accuracy, CNB and the ensembles were outperformed by S-Dummy $> 50\%$ of the concept drift-windows. Also, all ensembles but E-LRC have a lower average 'local minimum' accuracy than the S-Dummy.

Another notable result concerning the S-Dummy is that it has a higher local minimum accuracy than all other models (including RF) during the following concept drift-windows:

- **Session 12**: Epoch 8-11, 37-40.
- **Session 13**: Epoch 47-50.

Likewise, S-Dummy has the highest average accuracy during the following windows:

- **Session 3**: Epoch 14-18.
- **Session 9**: Epoch 12-18.
- **Session 11**: Epoch 10-13.
- **Session 12**: Epoch 8-11, 37-40.
- **Session 13**: Epoch 47-50.
- **Session 14**: Epoch 3-4, 12-17.

The S-Dummy also outperforms all models except for HTC (and RF) in terms of average accuracy. Regarding both local minimum accuracy and average accuracy, the only time that HTC is outperformed by the S-Dummy is when the S-Dummy outperforms RF. Additionally, out of 35 instances, HTC has a higher local minimum and average accuracy than RF 25 and 24 times respectively.

Concept Drift **Local minimum**	**GNB** ***Min. acc.***	**CNB** ***Min. acc.***	**HTC** ***Min. acc.***	**E-Dummy** ***Min. acc***	**E-LRC** ***Min. acc.***	**E-DT** ***Min. acc.***	**E-KNN** ***Min. acc.***	**S-Dummy** ***Min. acc.***	**RF** ***Min. acc.***
Session 1 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 1-3 (1, 2, 3)	3.73% ^	4.49% ^	7.96% ^	0.00%	0.78%	0.76%	2.32%	0.00%	3.12%
Epoch 14-15 (15)	21.32%	14.25%	27.01% ^	12.67%	24.01%	23.83%	24.10%	12.67%	24.23%
Session 2 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 8-11 (9)	31.65%	39.99% ^	19.59%	0.00% v	0.00% v	0.00% v	0.00% v	7.86%	37.62%
Epoch 18-22 (21)	50.13% ^	8.11%	21.96%	3.96%	3.96%	3.96%	3.96%	3.96%	23.74%
Epoch 40-42 (42)	18.69% ^	13.19% v	28.84% ^	13.57%	13.57%	13.57%	13.57%	13.57%	18.60%
Epoch 52-56 (52, 56)	14.67%	3.43%	34.05% ^	0.00%	0.00%	0.00%	0.00%	0.00%	17.23%
Session 3 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 1-4 (4)	28.46% ^	24.75%	29.72% ^	7.69%	33.54% ^	31.32% ^	14.70%	0.00%	27.98%
Epoch 10-12 (11, 12)	11.13%	6.26% v	11.50% ^	4.23% v	10.40%	9.84% v	8.28% v	10.30%	11.22%
Epoch 14-18 (15, 18)	10.69% v	7.18% v	20.00% ^	0.00% v	16.20% ^	12.12% ^ v	15.79% ^	12.64% ^	11.00% v
Epoch 31-36 (33, 36)	39.50% ^	6.26% v	38.51%	11.77%	13.71%	11.77%	11.77%	11.77%	39.32%
Session 4 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 1-3 (2, 3)	2.92%	1.12%	39.59% ^	0.00%	0.00%	0.00%	0.00%	0.00%	3.85%
Epoch 12-15 (12, 15)	32.07% ^	6.03%	22.83%	4.05%	10.14%	3.83% v	4.67%	4.05%	29.08%
Epoch 19-24 (20, 22, 24)	12.54%	5.24%	14.61%	0.00%	12.89%	12.04%	15.47%	0.00%	16.76%
Epoch 33-34 (34)	34.50%	22.16%	34.65%	13.74%	14.41%	13.79%	13.74%	13.74%	41.21%
Epoch 41-43 (42, 43)	38.72% ^	10.02%	34.14%	2.05%	11.19%	2.40%	2.69%	2.05%	36.91%
Session 7 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 24-25 (25)	28.21% ^	27.39% ^	26.23% ^	6.95%	6.95%	6.95%	6.95%	6.95%	21.51%
Session 8 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 7-10 (10)	20.87%	12.42%	36.36% ^	7.87%	13.32%	13.89%	14.28%	7.87%	35.08%
Epoch 16-19 (19)	38.88% ^	24.82%	39.00% ^	0.00% v	37.69% ^	37.30% ^	36.84% ^	0.52%	36.39%
Epoch 26-29 (29)	47.47%	9.30% v	56.14% ^	29.49%	31.16%	29.49%	29.49%	29.49%	53.72%
Epoch 34-36 (36)	73.87% ^	14.41%	21.55%	4.00%	4.92%	4.00%	4.00%	4.00%	63.87%
Epoch 53-56 (54, 55)	32.13%	12.61% v	32.24%	12.14% v	35.71% ^	13.01% v	17.21% v	17.86%	32.39%
Session 9 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 12-18 (12, 16, 17)	2.46% v	1.82% v	19.37% ^	0.00% v	0.10% v	0.10% v	0.23% v	13.34% ^	10.55% v
Epoch 28-29 (29)	15.85% ^	0.22% v	14.57% ^	14.50% ^	14.21% ^ v	14.50% ^	13.47% ^ v	14.50% ^	12.33% v
Epoch 51-54 (54)	49.60% ^	12.11% v	41.93%	20.21%	21.69%	20.47%	20.28%	20.21%	49.09%
Session 10 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 4-9 (5, 8)	31.40% ^	13.30% v	28.73% ^	21.70% v	20.76% v	20.80% v	20.14% v	22.84% ^	22.71% v
Epoch 12-15 (13, 14)	41.89% ^	33.90% ^	32.66% ^	12.87%	16.36%	16.42%	16.39%	12.87%	30.80%
Session 11 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 10-13 (11)	24.61%	25.55%	33.32% ^	24.19% v	24.19% v	24.19% v	24.19% v	24.30%	27.02%
Session 12 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 8-11 (10)	17.01% v	22.30% ^ v	24.48% ^ v	18.51% v	18.51% v	18.51% v	18.51% v	36.19% ^	21.61% v
Epoch 30-31 (31)	21.95% v	15.72% v	28.77% ^	25.46%	25.46%	25.46%	25.46%	25.46%	26.32%
Epoch 37-40 (38, 39)	0.14% v	0.00% v	12.04% ^ v	0.00% v	0.00% v	0.27% v	0.23% v	21.85% ^	6.12% v
Session 13 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 12-16 (13, 15)	11.21% v	33.48%	35.39% ^	11.12% v	10.99% v	11.25% v	11.34% v	15.17%	34.24%
Epoch 47-50 (47, 50)	4.93% v	23.05% ^ v	22.93% ^ v	5.40% v	2.49% v	3.52% v	1.78% v	26.29% ^	12.58% v
Session 14 *Min. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 3-4 (3)	21.84% v	0.00% v	31.95% ^	0.00% v	0.00% v	0.00% v	0.00% v	30.69%	30.69%
Epoch 12-17 (12, 16)	1.94% v	0.00% v	27.27% ^	1.27% v	1.27% v	13.01% v	11.25% v	14.89%	15.77%
Epoch 27-28 (28)	4.07% v	0.11% v	14.30% ^	9.25%	9.14% v	9.46%	9.46%	9.25%	10.11%
Avg. over sessions	24.03%	13.00%	27.55% ^	8.53% v	13.13%	12.24% v	11.79% v	12.78%	25.56%

Table 4.9: Concept drift: Local minimum accuracy. Local minimum accuracies for each of the models for each of the concept drift-windows. See Section 4.1.1 and 4.1.1.1 for further descriptions on how to interpret the table.

Concept Drift *Avg. over time window*	**GNB** *Avg. acc.*	**CNB** *Avg. acc.*	**Hoeff** *Avg. acc.*	**E-Dummy** *Avg. acc.*	**E-LRC** *Avg. acc.*	**E-DT** *Avg. acc.*	**E-KNN** *Avg. acc.*	**S-Dummy** *Avg. acc.*	**RF** *Avg. acc.*
Session 1 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 1-3 (1, 2, 3)	27.40% ^	14.98% v	23.83% ^	0.00% v	1.81% v	1.78% v	3.24% v	20.80%	20.99%
Epoch 14-15 (15)	41.50%	37.34%	44.09% ^	36.54%	42.73%	43.13% ^	43.15% ^	36.54%	43.11%
Session 2 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 8-11 (9)	67.99% ^	64.24%	68.63% ^	26.72% v	26.53% v	26.72% v	26.72% v	57.82%	65.07%
Epoch 18-22 (21)	68.80% ^	55.17% v	66.78% ^ v	67.55% ^	68.16% ^	61.08% v	63.89% v	67.55% ^	64.05% v
Epoch 40-42 (42)	54.15% v	55.97% ^ v	58.64% ^	56.40% ^	56.40% ^	56.40% ^	56.40% ^	56.40% ^	55.26% v
Epoch 52-56 (52, 56)	38.22% ^	16.46% v	51.20% ^	20.35% v	26.47% v	22.52% v	22.30% v	35.43%	36.62%
Session 3 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 1-4 (4)	48.12%	54.02% ^	49.35% ^	39.27%	49.48% ^	47.75%	39.53%	37.35%	48.64%
Epoch 10-12 (11, 12)	46.99%	38.65%	45.64%	32.81% v	48.48% ^	47.31% ^	45.40%	34.83%	47.22%
Epoch 14-18 (15, 18)	26.59% v	15.24% v	26.91% v	34.23% ^ v	32.22% ^ v	26.13% v	27.26% v	35.87% ^	28.11% v
Epoch 31-36 (33, 36)	52.95%	32.58% v	51.36%	46.93%	46.67% v	46.93%	46.91%	46.89%	54.07%
Session 4 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 1-3 (2, 3)	20.43%	19.68%	49.14% ^	15.76% v	15.76% v	15.76% v	15.76% v	19.46%	22.63%
Epoch 12-15 (12, 15)	51.85% ^	32.56%	47.72% ^	24.80% v	28.80% v	25.65% v	26.14% v	29.71%	47.49%
Epoch 19-24 (20, 22, 24)	43.47% ^	29.09%	42.21%	24.01%	38.22%	33.58%	36.03%	24.01%	42.96%
Epoch 33-34 (34)	46.27%	45.85%	47.51%	40.50%	41.42%	40.67%	40.38% v	40.50%	48.90%
Epoch 41-43 (42, 43)	43.22% ^	25.70%	42.35%	17.28%	28.18%	24.63%	26.28%	17.28%	43.01%
Session 7 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 24-25 (25)	42.12% ^	32.53%	41.07% ^	31.49%	31.49%	31.49%	31.49%	31.49%	38.66%
Session 8 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 7-10 (10)	49.54%	47.21%	54.26%	42.46%	49.02%	49.14%	49.91%	42.46%	56.51%
Epoch 16-19 (19)	52.99% ^	36.82%	54.15% ^	35.71% v	48.04%	43.09%	46.92%	35.84%	52.87%
Epoch 26-29 (29)	62.48% v	27.31% v	65.93% ^	64.04%	63.89% v	64.04%	64.05%	64.04%	65.37%
Epoch 34-36 (36)	85.90% ^	56.56% v	69.77%	64.41%	64.72%	64.41%	64.41%	64.41%	81.63%
Epoch 53-56 (54, 55)	59.52%	34.73% v	60.05%	50.46% v	54.94%	42.91% v	48.51% v	51.89%	61.44%
Session 9 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 12-18 (12,16, 17)	31.08% v	32.44% v	41.20% ^ v	35.80% v	35.53% v	40.05% ^ v	36.47% v	55.42% ^	37.55% v
Epoch 28-29 (29)	16.86% v	0.29% v	47.83% ^ v	49.90% ^	49.50% ^ v	49.47% ^ v	45.66% ^ v	49.90% ^	41.70% v
Epoch 51-54 (54)	61.63% ^	34.11% v	57.99%	54.09%	54.49%	54.17%	54.15%	54.09%	60.29%
Session 10 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 4-9 (5, 8)	45.34%	40.63%	42.81%	35.97% v	43.36%	37.62% v	43.26%	40.49%	45.80%
Epoch 12-15 (13, 14)	54.05% ^	46.59%	49.36% ^	28.59% v	42.32% v	42.03% v	42.68% v	44.11%	47.05%
Session 11 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 10-13 (11)	35.57% v	43.66% ^ v	41.00% ^ v	35.35% v	35.35% v	35.35% v	35.35% v	50.13% ^	36.47% v
Session 12 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 8-11 (10)	50.92% v	42.57% v	59.29% ^ v	54.47% v	54.47% v	54.47% v	54.47% v	64.38% ^	56.19% v
Epoch 30-31 (31)	50.55% v	24.02% v	57.51% ^	56.94% ^	56.94% ^	56.94% ^	56.94% ^	56.94% ^	52.51% v
Epoch 37-40 (38, 39)	20.49% v	9.04% v	40.29% ^ v	22.06% v	22.06% v	22.12% v	22.11% v	55.66% ^	37.19% v
Session 13 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 12-16 (13, 15)	46.38% v	51.05%	52.01% ^	46.30% v	45.65% v	46.00% v	46.61% v	46.72%	51.19%
Epoch 47-50 (47, 50)	21.56% v	38.37% ^ v	30.23% ^ v	21.54% v	15.66% v	15.94% v	15.41% v	47.13% ^	25.43% v
Session 14 *Avg. acc.*	*GNB*	*CNB*	*Hoeff*	*E-Dummy*	*E-LRC*	*E-DT*	*E-KNN*	*S-Dummy*	*RF*
Epoch 3-4 (3)	26.46% v	15.35% v	53.90% ^ v	15.35% v	15.92% v	15.35% v	15.35% v	65.35% ^	44.66% v
Epoch 12-17 (12, 16)	37.44% v	38.53% ^ v	43.65% ^ v	45.66% ^ v	45.59% ^ v	51.19% ^ v	47.33% ^ v	52.10% ^	38.45% v
Epoch 27-28 (28)	6.82% v	1.67% v	16.43% ^ v	33.43% ^	17.44% ^ v	16.89% ^ v	18.62% ^ v	33.43% ^	13.02% v
Avg. over sessions	43.88% v	34.03% v	48.40% ^	37.35% v	39.93% v	38.72% v	38.83% v	44.75%	46.06%

Table 4.10: Concept drift: Average accuracy. Average accuracies for each of the models for each of the concept drift-windows. See Section 4.1.1 and 4.1.1.1 for further descriptions on how to interpret the table.

Summary Concept Drift Local minimum	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN
Best performer	28.57% (10/35)	5.71% (2/35)	57.14% (20/35)	0.00% (0/35)	5.71% (2/35)	0.00% (0/35)	2.86% (1/35)
Best performer & in close proximity (<= 1%) to the best	37.14% (13/35)	8.57% (3/35)	65.71% (23/35)	0.00% (0/35)	5.71% (2/35)	0.00% (0/35)	2.86% (1/35)
Outperforming RF (^)	40.00% (14/35)	17.14% (6/35)	71.43% (25/35)	2.86% (1/35)	14.29% (5/35)	11.43% (4/35)	8.57% (3/35)
Outperformed by S-Dummy (v)	28.57% (10/35)	48.57% (17/35)	8.57% (3/35)	40.00% (14/35)	34.29% (12/35)	40.00% (14/35)	37.14% (13/35)

Table 4.11: Concept drift: Summary Local minimum accuracy performance. Summary of each model's performance based on results stored in Table 4.9. See Section 4.1.2 for further descriptions on how to interpret the table.

Summary Concept Drift Avg. over time window	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN
Best performer	31.43% (11/35)	8.57% (3/35)	45.71% (16/35)	8.57% (3/35)	2.86% (1/35)	2.86% (1/35)	0.00% (0/35)
Best performer & in close proximity (<= 1%) to the best	37.14% (13/35)	11.43% (4/35)	48.57% (17/35)	11.43% (4/35)	11.43% (4/35)	11.43% (4/35)	5.71% (2/35)
Outperforming RF (^)	31.43% (11/35)	14.29% (5/35)	68.57% (24/35)	20.00% (7/35)	25.71% (9/35)	22.86% (8/35)	17.14% (6/35)
Outperformed by S-Dummy (v)	40.00% (14/35)	57.14% (20/35)	31.43% (11/35)	54.29% (19/35)	54.29% (19/35)	57.14% (20/35)	57.14% (20/35)

Table 4.12: Concept drift: Summary Average accuracy performance. Summary of each model's performance based on results stored in Table 4.10. See Section 4.1.2 for further descriptions on how to interpret the table.

4.6 Analysis of Handling Six RSC Simultaneously

Another interesting area of analysis is when there is no clear concept drift, but there are several RSCs present simultaneously, which could potentially present a challenging period for the models to handle. An analysis examined a short period of four epochs in session 4, where all six RSCs were present. Figure A.5 and 4.4 illustrate the overall data distribution for the entire session, with the analyzed section highlighted in red, at epochs 5 - 8. The RSC distribution of the analyzed section can be seen in Table 4.13. The analysis was conducted by selecting the minimum value and calculating the average of the metrics accuracy, recall, precision and f1-score throughout the four epochs. The analysis was conducted with a 10-second epoch basis, indicating that E-DT and E-LRC did not employ an epoch time of 2 seconds, but were implemented with a 10-second epoch, for comparative purposes. The results can be seen in Table 4.14.

RSC	**Grey ice**	**Slush**	**Snow**	**Wet**	**Damp**	**Dry**	**Total**
Data points	1632	629	381	4161	3104	391	10 298
Percentage %	15.84	6.11	3.7	40.41	30.14	3.8	100.0

Table 4.13: Distribution and percentage of data points by RSC during epochs 5 to 8 in session 4

The best-performing model considering average accuracy is the E-LRC. It outperforms RF with 6.18%, and the second-best performing model, E-KNN with 5.33%. E-LRC, E-KNN and HTC all outperform RF indicating strength, additionally, HTC holds the highest min value, outperforming RF with 3.63% and the second-best E-LRC with 4.72%. The same pattern can be seen regarding the average recall since being identical [31].

Regarding the average precision and F1-score, HTC outperforms all models, achieving 1.29% above RF in precision, and a 0.34% higher F1-score. A notable strong performer regarding precision is GNB which is 0.3% above RF, and close for the lead. E-KNN also shows the highest min value regarding precision, with HTC and GNB close behind. A notable poor performer regarding all metrics in this analysis is CNB. It is outperformed by the S-Dummy in each average metric except precision. CNB also shows low minimum values for all metrics.

Accuracy	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN	S-Dummy	RF
Avg Accuracy	31.74%	9.46% v	39.08% ^	37.85%	45.07% ^	31.22%	39.74% ^	22.89%	38.89%
Min Accuracy	26.67%	0.29% v	33.31% ^	16.97%	28.59%	18.55%	25.06%	1.95%	29.68%

Precision	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN	S-Dummy	RF
Avg Precision	38.19% ^	25.13%	39.18% ^	17.17%	34.65%	30.25%	34.51%	7.82%	37.89%
Min Precision	19.42%	7.51%	19.68%	2.88%	14.09%	15.21%	20.02%	0.04%	25.61%

Recall	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN	S-Dummy	RF
Avg Recall	31.74%	9.46% v	39.08% ^	37.85%	45.07% ^	31.22%	39.74% ^	22.89%	38.89%
Min Recall	26.67%	0.29% v	33.31% ^	16.97%	28.59%	18.55%	25.06%	1.95%	29.68%

F1-score	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN	S-Dummy	RF
Avg F1-score	31.93%	6.47% v	36.68% ^	22.99%	35.83%	27.00%	34.67%	11.32%	36.34%
Min F1-score	20.59%	0.56%	22.60%	4.92%	18.57%	9.65%	17.20%	0.07%	27.80%

Table 4.14: Average and Minimum Metrics During E 5-8 in S4. Displaying the average and minimum values of the metrics accuracy, precision, recall and F1-score during the epochs 5 to 8 in session 4. See Section 4.1.1 for descriptions of how to interpret the table.

5
Discussion

This project's aim was to evaluate and identify ML models' strengths and weaknesses in relation to the provided dataset and provide a foundation for further research and development. Therefore, this chapter will discuss the results presented in the previous chapter, highlighting the strengths and weaknesses of the models and providing a concluding discussion and model recommendation. This chapter will further discuss the shortcomings of the method and the dataset, which could have affected the models' performances. Since being a project related to traffic safety, it is important to consider a model's performance across various scenarios.

5.1 Discussion about the Average Metrics

This project evaluated seven ML models divided into two groups, the standalone models and the ensembles. The two different approaches were tested to identify their respective strengths and weaknesses. The results show that there is no clear group that is outperforming the other. Regarding the average accuracy (and recall) shown in Tables 4.1 and 4.2 E-LRC and E-DT are the best performing models of all the evaluated. However, the ensembles E-KNN and E-Dummy are not performing better than the standalone models. HTC has a higher total average accuracy than E-KNN, and E-Dummy, suggesting that using an ensemble approach solely is not always the most effective approach, indicating there is more to consider. A question that arises is, what sets E-LRC and E-DT apart from the other ensembles? Except for having their unique classification methods, they are using a sliding window with a 2-second epoch time. This sliding window may influence their results since a shorter epoch time allows the ensemble model to update the ensemble faster with potentially more accurate classifiers. This introduces a bias towards more recent data. However, E-Dummy and E-KNN were both run with the same options during the hyperparameter tuning as E-LRC and E-DT but performed more effectively without the use of a sliding window and 2-second epoch time. This supports the idea that E-LRC and E-DT's performance is due to their unique classifying method rather than their use of a shorter epoch time. Although, there are other aspects to consider, both concerning the difference in epoch time and the tuning of hyperparameters using the hyperparameter set. These are discussed further in Section 5.3 and 5.7.

Regarding average precision, E-DT and E-LRC show poorer performance than their performance in accuracy, while HTC, GNB and CNB emerge to show some strengths, as seen in Table 4.3. This finding is likely a result of the trade-off between recall (accuracy) and precision as discussed in [4]. A higher recall often leads to a lower precision, and vice versa. It is thus important to note that the results from average accuracy must be considered in relation to the results of average precision. The same applies to recall since

in this project, weighted recall and accuracy are the same value. When considering this relationship, the dominance of E-LRC and E-DT is not that prominent, giving HTC and GNB somewhat more prominence. CNB is admittedly winning two sessions in average precision but has overall significantly lower average accuracy than GNB and HTC. When finally considering the average F1-score which takes both precision and recall (accuracy) into account, giving a more balanced assessment of the models' performances, it becomes clear that the most promising models are, in descending order, E-LRC, E-DT and HTC.

The models were all assessed against the baseline models, S-Dummy and RF. Regarding total average accuracy (and recall), S-Dummy is surprisingly ranked in 4th place, outperforming RF, which comes in 6th place, as seen in Table 4.1 and 4.2. S-Dummy also performs better than E-Dummy in 12 out of 14 sessions, which can be explained by the E-Dummy building up an ensemble of Dummy Classifiers trained on data stretching up to 20 epochs back in time, making the E-Dummy less flexible to changes. Furthermore, the S-Dummy outperformed most of the models regarding average accuracy (and recall) in sessions 7, 9, 11, 12, 13, and 14. During session 14, it outperformed all the other models with a total average of 69.79%, while E-LRC came in second with 68.32%. The characteristics of these sessions are that a condition is dominating for long periods and that the session contains relatively few concept drift-events and windows. Most of the concept drift-events are also happening relatively fast, which allows a new RSC to become dominant quickly. As a reference, S-Dummy shows a quite low average accuracy in session 4, which contains 10 concept drift-events and fewer periods being dominated by one RSC. The performance of the S-Dummy suggests that it does not pay off to train a more sophisticated model on the provided features. Instead, it appears that sometimes it is more efficient to base predictions solely on the previously observed condition. This might be true for some sessions and if only regarding accuracy. When considering the F1-score, the S-Dummy's performance is significantly lower. It reaches a total average F1-score of 53.90% in relation to the winning E-LRC at 64.61%, as seen in Table 4.4. Altogether, this does not indicate that the S-Dummy is a high performer, but rather that the others have even poorer average performances in these sessions.

The anticipated top-level baseline set by the RF model demonstrated relatively poor performance and arguably should not be considered top-level. Regarding average accuracy, RF is outperformed in 46 out of 98 cases (46.94 %) by the evaluated models and in 6 sessions out of 14 (42.86 %) by the anticipated low-level baseline set by S-Dummy. The conclusion is that the appointed top-level baseline for this project does not represent a high standard and outperforming this model does not necessarily indicate a strong performance. In hindsight, a more suitable alternative would have been to train RF on a larger data set such as the hyperparameter set, before evaluating the epochs in the assessment set. Such an approach may have resulted in a top-level baseline of higher performance.

5.2 Discussion on Handling Concept Drifts

A valuable feature of a model predicting RSC is to handle concept drifts since the model should be able to detect a sudden shift in RSC, especially in case the shift is towards an RSC that poses a potential hazard to traffic safety, such as grey ice. As accounted for in Section 1.4 (Limitations), this project does not involve evaluating how the model handles such categories that pose a higher risk if misclassified. However, a model that can handle concept drifts, should potentially handle whichever class the RSC shifts towards. The model's ability to identify and adapt to evolving RSC, should enhance the overall traffic safety of a vehicle.

To ensure diversity, the concept drift-events identified include a shift toward each of the possible classes in this project, including grey ice. The project evaluates two types of accuracies: local minimum accuracy and average accuracy during a concept drift-window. As described in Section 3.6.5.2 this project focuses on concept drift-windows instead of solely an epoch corresponding to a concept drift-event, since the minimum accuracy due to a concept drift does not necessarily occur at the time of the event. In addition, evaluating the average during such concept drift-windows was an attempted approach to assess a model's ability to recover after a concept drift, since this is a sought-after property [7]. A model with a long recovery time should have a low average accuracy during the concept drift-window, whereas a short recovery time should correspond to a higher average accuracy. This correlation is the probable cause of S-Dummy outperforming nearly all models, HTC and RF excepted, in terms of average accuracy averaged over sessions. During the concept drift-event the S-Dummy will train a Dummy Classifier that always predicts that class that is in the majority during the event. Such a majority class is commonly the majority for some epochs following the event. When such circumstances arise, the S-Dummy will have an accuracy equal to the proportion of the majority class during those epochs. Given that average accuracies typically fall below 50%, epochs characterized by accuracies surpassing the 50% threshold will positively influence a model's overall accuracy. However, the impact of the S-Dummy consistently predicting the majority class of the former epoch causes the S-Dummy to perform poorly concerning local minimum accuracy during many of the concept drift-windows. Yet, there are exceptions. During three of the windows, the S-Dummy has the highest local minimum accuracy of all models (including RF). In all such cases, the class that constituted the majority prior to the concept drift-event, is of a higher proportion during the event than all the other models' accuracies. An example of this is the concept drift-window "session 12, epoch 8 - 11", where a concept drift-event occurs at epoch 10 (see Figure A.13 in Appendix). At epoch 10, the former majority class "Dry" constitutes 36.19 % (equal to S-Dummys accuracy) of the data points, whereas the local minimum accuracies of the other models are below that. So even though the S-Dummy was implemented as a low-level baseline, being a model biased towards processed data of the prior epoch is in some situations beneficial, especially when comparing its performance to that of the other models. Still, in cases when S-Dummy outperforms the other models, it may rather be an indication of the other models performing poorly, since S-Dummy will not predict a single instance of the majority class correctly during a concept drift-event. Also, an ideal model should achieve an accuracy of 100% in all types of situations. None of the models come close to such accuracies.

Another thing to bear in mind is that the class that becomes the majority during a concept drift event is often present to some extent in the epochs preceding the event. This likely influences the accuracy of the models. Also, the models' potential of remembering patterns learned from data further back will affect how they handle a concept drift. For example, HTC is not rebuilt and therefore remembers patterns from all previously seen data, which is a potential reason why HTC outperforms all other models in all investigated scenarios involving concept drifts. The ensemble models, on the other hand, are limited to remembering from a number of training sets equal to the size of the ensemble and likely have a high proportion of classifiers biased toward frequently occurring classes of the last epochs. By definition, the class that becomes the majority has not been in the majority for at least five epochs, so models that are not as biased towards the last epochs should perform better. There are several methods for improving ensemble models' ability to handle concept drifts [18, 7]. For example, this project uses accuracy to evaluate when to switch a classifier to another. However, it has been shown that accuracy is not the most optimal metric to evaluate such scenario [18] and there are other metrics available [7] which would potentially improve the ensemble models' performances in handling concept drifts.

Two models' performances that have not yet been discussed are GNB and CNB. Even though the probability of a class is affected by the distribution during the last epochs, GNB and CNB do not forget classes and may therefore be more equipped to handle a concept drift where a class not seen for many epochs becomes the majority. This may be a contributing factor to why GNB came in second place in performance related to concept drifts. A bit surprisingly, especially given that the model is particularly well-suited to handle imbalanced data sets, CNB had a poor performance. Whether this is due to the model itself or to a slightly different preprocessing procedure (including scaling) is speculative. For a further discussion, see Section 5.4.

5.3 Discussion on Handling Six RSC Simultaneously

An interesting area of analysis is when there are several RSCs present simultaneously. Such a scenario could potentially present a challenging period for the models to handle, and preserve traffic safety through high accuracy. In the results from the analysis of simultaneously handling six RSC conditions, it is notable that although E-LRC and HTC performed best among the evaluated models, their metrics value for accuracy, precision, recall and F1-score were significantly lower than their average values across the entire session, as can be seen in Table 5.1. This fact applies to all the models and indicates they all struggle to adapt. E-Dummy shows the smallest drop in all metrics except for the F1-score when compared to its overall averages during session 4. In terms of F1-score, E-KNN is the model that shows the smallest drop. However, E-Dummy's low decrease across the metrics, should not be considered a strong performance due to its significantly lower average values for these metrics during session 4, as seen in Tables 4.1, 4.2, 4.3 and 4.4. Similarly, although E-KNN has the least decrease in F1-score, it has a relatively low average F1-score over the session. A notable fact is also CNB's large decrease in all metrics, even though having overall low average values to begin with. This analysis concludes that rapidly changing and simultaneous RSCs present a weakness for all models and pose a potential risk regarding traffic safety, which aligns with the conclusion from the analysis of concept drift. The best-performing model during the analyzed period in

terms of accuracy managed to achieve an average of 45.07%, which cannot be considered a strong or safe performance.

Decrease (Percentage points)	GNB	CNB	HTC	E-Dummy	E-LRC	E-DT	E-KNN	S-Dummy	RF
Decrease in Accuracy	- 26.08	- 39.23	- 21.90	- 13.94	- 18.17	- 32.07	- 15.17	- 28.66	- 19.18
Decrease in Precision	- 25.78	- 37.48	- 26.18	- 23.39	- 30.25	- 32.95	- 24.72	- 31.81	- 27.84
Decrease in Recall	- 26.08	- 39.23	- 21.90	- 13.94	- 18.17	- 32.07%	- 15.17	- 28.66	- 19.18
Decrease in F1-score	- 25.81	- 38.85	-23.41	- 23.96	- 24.42	- 33.10	- 16.17	- 31.99	- 21.93

Table 5.1: Decrease from Average Metrics During E5-8 in S4. Showing the difference between the average metric values during session 4 and the calculated metric values during epochs 5-8 in session 4 for each model respectively. The orange colour indicates the smallest decrease.

It is important to note that this analysis was conducted on a single time frame during one session, which can lead to difficulties in drawing clear conclusions. However, the results align quite well with the other analyses in this report. The notable difference is E-DT relatively poor performance during this scenario in relation to its overall strong average performance over all sessions.

5.4 Discussion on CNB's Low Performance and Potential Scaling Issues

Throughout the analysis of the models, CNB has a persistently poor performance with merely a few exceptions. It could be the case that the CNBs classification method is unsuitable for the provided data, which is contradicted by CNB being particularly suited for imbalanced data sets. Another possible contributing factor is a slightly different preprocessing procedure. CNB was unable to handle negative features due to implementation limitations, as discussed in Section 3.1.1, why feature values were scaled before training the model. It is thus important to note this difference in preprocessing as it may have affected the performance of the model. To ensure that the difference entails a significant difference in performance or not, this project should have scaled the features for all the models to be able to conduct more uniform comparisons. This was not carried out, which should be considered a weakness of the overall method.

5.5 Discussion on Using 10 Seconds Epoch Time for E-LRC and E-DT

In the analysis of concept drift and the simultaneous handling of six RSCs, the generated metric values of the models E-LRC and E-DT using sliding windows with a 2-second epoch time, were not used. This implies that the training sets did not overlap as described in the training procedure 3.4.2. The underlying reason was to facilitate comparison between models. Instead, metric values from E-LRC and E-DT running with no sliding window and a 10-second epoch time were utilized. However, this results in unfair comparisons given that the sliding window was a hyperparameter for the ensemble models and the strongest performing option for E-DT and E-LRC during hyperparameter tuning. These models might underperform without it. A more fair comparison would possibly be to analyze the models running with their strongest configuration as determined during the hyperparameter tuning. A problem regarding this approach would be selecting a starting epoch for the analysis. Choosing epoch 5 with a 10-second epoch time would correspond to epoch 25 with a 2-second epoch time. However, the output metrics at that point would align with different time frames being assessed. With a 10-second epoch time, the entire preceding epoch of 10 seconds is assessed, while with 2 seconds, only the previous 2 seconds. One possible solution to this problem could involve redefining the starting point of the analyzed time frame to include epochs 21 to 25. This adjustment will cover the same time frame and data points as starting at epoch 5 with a 10-second epoch time, although partitioned into several smaller-sized sets. This might give a fairer comparison since the same number of data points is being assessed. However, it does not solve the issue that differently sized subsets will contain different data distributions leading to varied evaluations. This can impact model performance and thus not provide a consistent basis for comparison between models. Since this comparison not being entirely ideal either, the choice became to compare the models running with a 10-second epoch time for convenience.

Regarding the analysis of the simultaneous handling of six RSCs, a comparison was made where output metrics from E-LRC and E-DT when using a sliding window with a 2-second epoch time were considered. The time frame for conducting the comparison is in 10-second epoch time between epochs 5 to 8, and in 2-second epoch time between epochs 21 to 40, following the idea discussed above. The comparison can be seen in Table 5.2. Notably, E-LRC shows a small drop in average accuracy when utilizing a 2-second epoch time in comparison to running with a 10-second epoch time. E-DT, on the other hand, improves in average accuracy when using a 2-second epoch time. A 2-second epoch time also improves the performance of both models in terms of average precision. However, for the average F1-score, E-LRC performance drops slightly when running with a 2-second epoch time, whereas E-DT performance improves. Notably, for both E-LRC and E-DT, the minimum values are significantly lower using the shorter epoch time, which could be the result of smaller epochs and more pronounced variations in the measurements. The key discovery from this comparison is that the strongest configuration of the ensembles does not necessarily result in a stronger performance when handling six RSCs simultaneously. E-LRC 2-S lost in accuracy, while E-DT 2-S achieved a substantial improvement. This finding both supports and contradicts the results from [18], which states that as the training sets for an ensemble get smaller, the performance tends to decrease. However,

the performance when using a 2-second epoch time will also be affected by training on overlapping training sets. Consequently, this further complicates the process of drawing conclusions from the analysis of comparing 2-second and 10-second epoch times.

Accuracy	**E-LRC 10-S**	**E-DT 10-S**	**E-LRC 2-S**	**E-DT 2-S**
Avg Accuracy	45.07%	31.22%	41.61%	42.48%
Min Accuracy	28.59%	18.55%	8.07%	7.02%

Precision	*E-LRC 10-S*	*E-DT 10-S*	*E-LRC 2-S*	*E-DT 2-S*
Avg Precision	34.65%	30.25%	42.70%	39.46%
Min Precision	14.09%	15.21%	2.46%	2.40%

Recall	*E-LRC 10-S*	*E-DT 10-S*	*E-LRC 2-S*	*E-DT 2-S*
Avg Recall	45.07%	31.22%	41.61%	42.48%
Min Recall	28.59%	18.55%	8.07%	7.02%

F1-score	*E-LRC 10-S*	*E-DT 10-S*	*E-LRC 2-S*	*E-DT 2-S*
Avg F1-score	35.83%	27.00%	35.32%	35.56%
Min F1-score	18.57%	9.65%	3.31%	2.52%

Table 5.2: Difference in average metrics of 11 and 2 second epoch time. Displaying the difference in average metric values for E-LRC and E-DT during epochs 5 to 8 in session 4 when running with 2-second versus 10-second epoch time. 10-S and 2-S indicate that the models were run with 10-second and 2-second epoch times, respectively.

5.6 Concluding Discussion of the Models: Strengths and Weaknesses

This project focuses on finding strengths and weaknesses of ML models designed to predict RSC, to serve as a foundation for the company Klimator. A strength can be assessed as how well a model performs in comparison with other models, or by achieving, for example, an accuracy above a certain percent. In terms of the former, a strength of HTC is its ability to handle concept drifts. However, higher local minimum accuracies during concept drifts would be preferable, HTC having an average local minimum accuracy of barely 27.55%. Such a model is not trustworthy if used in a vehicle, especially in the case of a sudden shift towards an RSC posing a potential hazard to traffic safety, such as grey ice. In comparison to the other models, E-LRC and E-DT are the strongest and second strongest models in terms of average performance, measured by the F1-score. Also in this context, higher values would be preferable, with these models averaging 64.61% (E-LRC) and 64.45% (E-DT) over sessions. However, they exhibit slightly higher values for the main metric evaluated in this project, with E-LRC averaging an accuracy of 68.81% and E-DT averaging 68.82% over sessions, both outperforming the other models. Still, achieving even higher accuracy values is desirable, as both models still get the prediction wrong in >31% of the cases.

A notable weakness of the ensemble models is their underperformance in handling concept drifts, which likely contributes to the average accuracy of E-LRC and E-DT not being higher. Presumably, a contributing factor is that the ensemble models are biased toward the RSC processed during the preceding epochs, giving lower priority to classes seen earlier during training, or even forgetting about such classes. As a result, previously learned patterns associated with the class that become the majority during the concept drift-event might be ignored. However, based on this logic, the S-Dummy classifier should be outperformed by all models due to its complete bias towards the majority class of the epoch preceding the event. This is not the case. For example, E-DT is outperformed by S-Dummy in 40.00% of all concept drifts in terms of local minimum accuracy. Since S-Dummy will not make a single correct prediction regarding the class that becomes the majority during the drift, it can be concluded that making accurate predictions for minority classes during a drift contributes to an increase in the average accuracy of the models. However, if the local minimum accuracy during concept drifts were solely dependent on making such predictions correctly, both E-DT and E-LRC should outperform HTC, since their performance in terms of average accuracies is superior. Although this is not the case, and the fact that HTC outperforms all other models suggests that the definition of concept drift used in this project reflects, to some extent, how well the model handles concept drift.

The fact that E-LRC and E-DT exhibit the best average performance but inferior performance during concept drift, combined with S-Dummy's good results, leads to the conclusion that, on average, it is beneficial for a model to be biased towards newly processed RSC. As previously discussed in Section 5.1, this is likely due to a condition being in a significant majority during many epochs. Another notable result is that HTC had the third-highest average accuracy (63.98%) and F1-score (63.14%) over sessions. This observation, along with HTC demonstrating the strongest performance in handling concept drifts, positions HTC as a potential candidate. However, E-LRC, E-DT, and HTC all

demonstrate suboptimal results when put into a traffic safety context, indicating the need for further optimization before they can be employed for predicting RSC in real-life scenarios.

5.6.1 Models: Recommendations and Further Development

Based on the strengths and weaknesses of the models, three possible candidates show potential: HTC, E-DT and E-LRC. There are several ways to optimize the three top candidates based on the results of this project. In the sessions of this project, the models achieve high accuracies (up to 100%) under epochs when a class is in a significant majority. However, in most of the sessions, data distribution varies continuously, affecting the performance of the models. There are ways to increase the models' response to such drifts, making them more adaptable to fluctuations. One solution would be to use a variant of HTC that can replace subtrees that are no longer accurate and thereby adapt to drifts without increasing its size. One such example is the Extremely Fast Hoeffding Adaptive Tree [53], which adapts quickly to drifts. Implementing such a version would enable indefinitely updating an HTC in response to drifts, without concerns about the tree becoming too large. Similarly, there are methods for enhancing the ensemble models' adaptation to drifts [7], which could be beneficial for E-DT and E-LRC, potentially increasing their accuracy during concept drifts. For example, by employing a different approach when deciding whether to replace one of the classifiers in the ensemble or using an alternative method to weigh the votes of ensemble members, instead of relying solely on accuracies from the previous test set. Making the models more equipped to handle drifts, would also be beneficial to the overall performance of the models.

Given that HTC was the strongest performer during concept drifts whereas E-LRC and E-DT outperformed in average performance, it would be an interesting scenario to use HTC in combination with one of the other two models. Detection mechanisms for identifying concept drifts exist [7]. By leveraging such a detection mechanism a model predicting RSC could transition, for instance, from E-LRC to HTC upon detecting a concept drift, and then switch back to E-LRC when the ensemble has adapted to the new concept. This method would of course need further research, but could be a potential way of utilizing the strengths of the different models.

5.7 Discussion on Method and Dataset

In addition to optimizing the models further, there are ways to improve this project, in order to produce even more reliable results. One such aspect concerns the data set. The data set utilized by this project contained 2 013 420 data points distributed across 20 sessions, taken during various times and weathers, as described in Section 3.1 and Table 3.1. One issue with the dataset is that the data points are not evenly distributed over time. This means that a 10-minute session and a 10-second epoch, both may contain different amounts of data points compared with another session and epoch of the same length, respectively. The sessions are also different in length, with 14 of them being approximately 10 minutes, while the remaining six are varying from 1.28 minutes to 8.12 minutes. Several sessions also contain gaps, where no data have been captured during a certain period of time, as can be seen in session 3, epoch 44, Figure A.4 and session 7,

epochs 8 - 11, Figure A.8. All these issues lead to challenges in comparing sessions and epochs against each other and might introduce bias toward data points in sessions and epochs of smaller sizes.

Another issue with the data set is the true label of the conditions. This project considered the determined RSC category by the laser as the true RSC, which emphasizes the importance of the laser making accurate determinations. If the laser was inconsistent in creating the true labels for the RSC, the models have been trained on features that have had ambiguous true labels, which can result in poorer performance. No estimation of the laser's accuracy could be provided by this project to assess the potential impact of this issue.

Regarding the partitioning of the dataset into the assessment set and the hyperparameter set, as discussed in Section 3.6.1, the split was based solely on the RSC distribution and not the number of concept drift-events or the session length. Since concept drift played a significant role in the analysis of the models' strengths and weaknesses, it may be the case that the partitioned hyperparameter set is not representative in this aspect. Another notable difference between the hyperparameter set and the assessment set is that two of the sessions in the hyperparameter set are significantly shorter than 10 minutes. As seen in Figure A.17 and A.18 the length is 1.28 minutes and 2.2 minutes. These issues may have resulted in choosing hyperparameter combinations perhaps not optimal for the assessment set.

The method for hyperparameter tuning was developed by this project as discussed in Section 3.6.2, which also covers the evaluation and the reliability of the method itself. Further discussion on this topic will not be provided here. Regarding the hyperparameter tuning process and the search for maximum performance, it is important to note that there is a possibility of existing several local maxima in addition to the global maximum, for a certain hyperparameter interval. For example, regarding the hyperparameter grace period belonging to HTC, as seen in Table A.1 this project used a search space between 50 and 350 during the first tuning iteration. The maximum performance was found when using 275 as a grace period value. It is thus possible that 275 is a local maximum, and that the global maximum is outside the initial search space. The search spaces were chosen around the default value, and in some cases based on search spaces or specific values used by previous studies. Due to time constraints and computing resources, larger search spaces and additional iterations were not possible.

Regarding the hyperparameter tuning process and the selection of parameters, this project selected some hyperparameter values that were significantly different from the default value. These were the smoothing values for GNB, the split confidence for HTC and the alpha value for CNB. The smoothing parameter belonging to GNB is by default 10^{-9}, and the selected value was 0.3. HTC's split confidence is by default 10^{-7}, and the selected value was 0.3. The alpha value belonging to CNB is by default 1.0, but the project selected the value 1000.0. It would have been interesting to compare the performance of these models when running with their default values against the performance of the models in this project, to further validate the hyperparameter tuning procedure. A hyperparameter that was arbitrarily chosen was max iterations belonging to LRC. The default value is 100, and as a response to warnings of non-convergence as discussed in Section 3.6.4.4, the

value was set to 1000. The choice of 1000 over a smaller value might have affected the efficiency of the model.

5.7.1 Methodology: Further Development

Considering the training, evaluation and tuning methods, several limitations had to be applied to complete the project within the given time frame. Additionally, some ideas for implementation emerged after the completion of the results and were thus not implemented. This section presents interesting proposals and ideas for others to test and develop further in future work.

One such idea is to use the epoch size as a hyperparameter and try to find an optimal size for the models, respectively. This is interesting since [7] show that different sizes can affect how well a model can recover from concept drift. It would also have been interesting to test different sizes of the sliding window during tuning, thus alternating the possible sizes of the training set.

Further interesting areas for future work could be to evaluate the models on datasets with longer time spans. This project evaluated dataset sessions with a length of approximately 10 minutes. The interesting aspect would be to evaluate how the performance of the models evolves over a longer period. Alternatively, customizing a data set for iterative training, simulating concept drift-events under controlled conditions. This approach would enhance the understanding of how the models handle shifts in RSC, by customizing the data set so that the measured performances strictly depend on the concept drift-event, excluding influence due to RSC previously encountered. Such an approach has previously been conducted, to evaluate how models handle concept drifts [18].

Concerning the analysis, this project focused on analyzing the models' performance on all the RSCs as a group and not one by one. It would have been valuable to understand more about how well the models predict each RSC independently, and how the accuracy is affected by different distributions, several RSC simultaneously and concept drifts. As seen by [54], the accuracy when classifying snow differs from the accuracy when classifying dry RSC. An investigation of this could result in finding models being strong or weak in predicting particular RSCs. Another interesting area of investigation could be to merge the two RSCs wet and damp since they are closely related to each other. This change might give the models an overall higher accuracy, potentially enhancing their reliability in various scenarios.

Regarding the training procedure. Both the standalone and ensemble models excluded any kind of pre-training, meaning that the first epoch of each session was the models' first exposure to RSC data, which could disproportionately influence the models' initial performance and shape their learning trajectory. An interesting area for future work could thus be evaluating the results from this study with models that have been pre-trained before starting incremental learning. This approach would potentially aid the models to faster reaching a high performance, but might also pose a risk regarding overfitting. Another interesting comparison that can be made is letting a suitable model train on a representative and well-distributed RSC dataset and then use it as a non-incremental

model and see how well it performs. This strategy would shed light on the effect of using incremental learning approaches versus not using them at all. This might serve as a new attempt at creating a low-level baseline.

A final thought is to question whether the provided features in the data set are optimal for training ML models. Since no investigation of their feasibility was conducted by this study, it could be interesting to investigate if the RSC could be represented by other potentially more optimal features, which might give better performance to the models. This thought emerged from observing the high relative performance conducted by the S-Dummy, which never trains on any features.

6
Conclusion

This project identifies certain strengths and weaknesses of different ML models in predicting RSC. A weakness that all evaluated models have in common is their limited ability to adapt to fluctuations and drifts in RSC distribution. This deficiency makes the models unfit for use in real-world applications, where RSC conditions can vary unpredictably. However, focusing on strengths based on model performance in comparison to the other models, three models stand out: HTC shows more potential than the others in terms of handling concept drifts, whereas E-DT and E-LRC have the highest average performance. These three models are recommended for future research and development as they demonstrate both potential and room for further optimization, which could lead to even better performances. Another notable finding is that in certain situations, a model making predictions based on which RSC was most frequent in recently processed data is preferable to making predictions based on the feature values utilized in this project. This suggests that there may be other feature values more suited for categorizing RSC in those scenarios. Nevertheless, our findings provide Klimator with a foundation for future research and development, in their pursuit to make a model fit for predicting RSC. Such a model has the potential to assist in autonomous vehicles and ensure traffic safety.

Bibliography

[1] Klimator., "Klimator,". [Online]. Available: https://www.klimator.se/ (accessed on: 2024-04-11).

[2] Y. Ma, M. Wang, Q. Feng, Z. He, and M. Tian, "Current Non-Contact Road Surface Condition Detection Schemes and Technical Challenges," *Sensors*, vol. 22, no. 24, pp. 9583, Dec. 2022, doi: https://doi.org/10.3390/s22249583.

[3] U. D. of Tranportation Federal HighwayAdministration, "How do weather events impact roads ?," Feb 2017. [Online]. Available: https://ops.fhwa.dot.gov/weather/q1_roadimpact.htm (accessed on: 2024-01-17).

[4] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, Machine Learning: A First Course for Engineers and Scientists. Cambridge: Cambridge University Press, 2022.

[5] V. Jacob, G. V. R. Sagar, K. Goura, and P. S. Subhashini Pedalanka, "Brain tumor classification based on deep CNN and modified butterfly optimization algorithm," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging and Visualization*, vol. 11, no. 6, pp. 2106-2117, Jun. 2023, doi: https://doi.org/10.1080/21681163.2023.2219754.

[6] P. Kulkarni, Reinforcement and systemic Machine Learning for decision making. New Jersey: Wiley-IEEE press, 2012.

[7] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132-156. 2017, doi: https://doi.org/10.1016/j.inffus.2017.02.004.

[8] Scikit-Learn., "sklearn.naive_bayes.GaussianNB,". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB (accessed on: 2024-02-29).

[9] Scikit-Learn., "Sklearn.naive_bayes.ComplementNB,". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.ComplementNB.html#sklearn.naive_bayes.ComplementNB (accessed on: 2024-03-14).

[10] A. Paszke, A. Chaurasia, S. Kim, E. Culurciello, "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation," *Cornell University*. [Online]. Jun. 2016. Available: http://arxiv.org/abs/1606.02147 (accessed on: 2024-01-19).

[11] D.D.C Maceda and J.C.D Cruz, "Rainfall Classification Model for the Philippines using Optimized K-nearest Neighbor Algorithm with GridSearchCV Hyperparameter Tuning," *2023 IEEE 13th International Conference on Control System, Computing and Engineering (ICCSCE)*. pp.51-55. 2023, doi:https://doi.org/10.1109/ICCSCE58721.2023.10237156

[12] Scikit-Learn., "sklearn.linear_model.LogisticRegression,". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression (accessed on: 2024-04-04).

[13] Scikit-Learn., "sklearn.tree.DecisionTreeClassifier,". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn-tree-decisiontreeclassifier (accessed on: 2024-03-14).

[14] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the KDD conference*, 2000, pp. 71–80.

[15] J. P. Barddal and F. Enembreck, "Learning Regularized Hoeffding Trees from Data Streams," *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 574-581, Aug. 2019, doi: https://doi.org/10.1145/3297280.3297334.

[16] Scikit-Learn., "1.9. Naive Bayes,". [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html (accessed on: 2024-04-04).

[17] J.D Rennie, et al. Tackling the poor assumptions of naive bayes text classifiers. in *ICML*. vol. 3, pp.616-623, 2003.

[18] W. N. Street and Y. Kim, "A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification," *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. pp. 377-382. 2001, doi:https://doi.org/10.1145/502512.502568.

[19] A. Dogan and D. Birant, "A Weighted Majority Voting Ensemble Approach for Classification," *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pp. 1-6. 2019, doi: 10.1109/UBMK.2019.8907028.

[20] Scikit-Learn., "Metrics and scoring: quantifying the quality of predictions,". [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score (accessed on: 2024-05-01).

[21] A.Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin,* vol. 106, no. 2, p. 58, 2004.

[22] H. He and E.A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, no. 9, pp. 1263– 1284, 2008.

[23] L. Baier, V. Kellner, N. Kühl, and G. Satzger, "Switching Scheme: A Novel Approach for Handling Incremental Concept Drift in Real-World Data Sets," in *Hawaii International Conference on System Sciences (HICSS)*, 2021, pp. 120-129, doi: https://doi.org/10.24251/HICSS.2021.120.

[24] J. Gama et al., "A Survey on Concept Drift Adaptation," ACM Computing Surveys, vol. 46, no. 4, April. 2014, doi: https://doi.org/10.1145/2523813.

[25] H. Meng, et al., "A survey of active and passive concept drift handling methods," *Computational Intelligence,* Vol. 38, no. 4, pp. 1492-1535, Aug. 2022, doi: https://doi.org/10.1111/coin.12520.

[26] J. C. Schlimmer, R.H. Granger Jr, "Incremental learning from noisy data," *Machine Learning* Vol. 1, pp. 317 - 354, Sep. 1986. [Online] Available: https://link.springer.com/article/10.1007/BF00116895

[27] Scikit-Learn., "6.3.Preprocessing data," [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html (accessed on: 2024-04-11).

[28] Python., "Python™,". [Online]. Available: https://www.python.org/downloads/ (accessed on: 2024-03-29).

[29] NumPy., "NumPy,". [Online]. Available: https://numpy.org/ (accessed on: 2024-03-29).

[30] Pandas., "Pandas,". [Online]. Available: https://pandas.pydata.org/ (accessed on: 2024-03-29).

[31] Scikit-Learn., "sklearn.metrics: Metrics,". [Online]. Available: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics (accessed on: 2024-03-29).

[32] Matplotlib., "Matplotlib: Visualization with Python,". [Online]. Available: https://matplotlib.org/ (accessed on: 2024-04-08).

[33] Scikit-Multiflow., "API Reference skmultiflow trees HoeffdingTreeClassifier," 2020. [Online]. Available: https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.trees.HoeffdingTreeClassifier.html (accessed on: 2024-02-02).

[34] Scikit-Learn., "sklearn.neighbors.KNeighborsClassifier,". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier (accessed on: 2024-03-21).

[35] Scikit-Learn., "sklearn.dummy.DummyClassifier," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html (accessed on: 2024-04-04).

[36] Scikit-Learn., "sklearn.ensemble.RandomForestClassifier,". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier (accessed on: 2024-03-22).

[37] L. Dong, X. Li, G. Xie, "Nonlinear Methodologies for Identifying Seismic Event and Nuclear Explosion Using Random Forest, Support Vector Machine, and Naive Bayes Classification," *Abstract and Applied Analysis*, vol. 2014, Article ID 459137, Feb. 2014. doi:https://doi.org/10.1155/2014/459137

[38] Y. Kim, Y. Kim, "Explainable heat-related mortality with random forest and SHapley Additive exPlanations (SHAP) models," *Sustainable Cities and Society*, vol. 79, article no. 103677, Apr. 2022. doi: https://doi.org/10.1016/j.scs.2022.103677

[39] X. Hu et al., "Estimating PM2.5 concentrations in the conterminous United States using the random forest approach" *Environmental Science & Technology*, vol 51, pp. 6936-6944, 2017.

[40] M. Mohammady, H.R. Pourghasemi, M. Amiri, "Land subsidence susceptibility assessment using random forest machine learning algorithm," *Environ Earth Sci*, vol. 78, no. 15, pp. 503, Aug. 2019. doi:https://doi.org/10.1007/s12665-019-8518-3

[41] G. James, D. Witten, T.Hastie, R. Tibshirani, *An Introduction to statistical learning with Applications in R*, New York, NY, USA: Springer, 2013.

[42] R. Soorya, A. N. T. Neenu, "Leveraging Machine Learning for Fraudulent Social Media Profile Detection," *Journal of Computational and Applied Information Technology,* vol. 24, nr. 1, pp. 118-136, 2024, doi: https://doi.org/10.2478/cait-2024-0007

[43] L. Wang and C. Wu, "Dynamic imbalanced business credit evaluation based on Learn++ with sliding time window and weight sampling and FCM with multiple kernels," *Information Sciences*, vol. 520, pp. 305-323. 2020, doi: https://doi.org/10.1016/j.ins.2020.02.011.

[44] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice", Neurocomputing, vol. 415, pp. 295-316, 2020, doi: https://doi.org/10.1016/j.neucom.2020.07.061

[45] M. M. Asiri, et al., "Short-Term Load Forecasting in Smart Grids Using Hybrid Deep Learning," *IEEE Access*, vol. 12, pp. 23504-23513, 2024, doi:/10.1109/ACCESS.2024.3358182.

[46] A.P. Noto, D.R.S Saputro, "Classification data mining with Laplacian Smoothing on Naïve Bayes method,"*AIP Conference. Proc.*, 28. November. 2022, vol. 2566, no. 030004 doi: https://doi.org/10.1063/5.0116519

[47] F. Arden, C. Safitri, "Hyperparameter Tuning Algorithm Comparison with Machine Learning Algorithms," in *Proc. IEEE 6th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Yogyakarta, Indonesia, 13-14 Dec. 2022, doi: https://doi.org/10.1109/ICITISEE57756.2022.10057630

[48] C. Dewi and R.C. Chen, "Complement Naive Bayes Classifier for Sentiment Analysis of Internet Movie Database", in *Proc. - Intelligent Information and Database Systems - 14th Asian Conference, ACIIDS 2022, Lecture Notes in Computer Science*, vol. 13757, pp 81-93. doi:https://doi.org/10.1007/978-3-031-21743-2_7.

[49] M. Khannouz, T . Glatard, "A Benchmark of Data Stream Classification for Human Activity Recognition on Connected Objects", Sensors(Basel)., vol. 20, no. 22, pp. 6486, Nov. 2020, doi: https://doi.org/10.3390/s20226486

[50] H. Zou, T. Hastie, "Regularization and variable selection via the elastic net". *Journal of the Royal Statistical Society Series B: Statistical Methodology,* Vol. 67. no. 5, pp. 301-320 Apr. 2005, doi: https://doi.org/10.1111/j.1467-9868.2005.00527.x

[51] Y. Liang, et al., "Surrogate modeling for long-term and high-resolution prediction of building thermal load with a metric-optimized KNN algorithm," *Energy and Built Environment,* Vol. 4, no. 6, pp. 709-724, Dec. 2023. doi: https://doi.org/10.1016/j.enbenv.2022.06.008

[52] P. Cunningham, S. J. Delany, "k-Nearest Neighbour Classifiers - A Tutorial," ACM Computing Surveys, vol. 54, no. 6, pp. 1-25, Jul. 2021, doi: https://doi.org/10.1145/3459665

[53] C. Manapragada, M. Salehi, G. I. Webb, "Extremely Fast Hoeffding Adaptive Tree," *2022 IEEE International Conference on Data Mining (ICDM)*, pp.319-328, 2022, doi: 10.1109/ICDM54844.2022.00042.

[54] H. Zhang, S. Azouigui, R. Sehab, M. Boukhnifer, "Near-infrared LED system to recognize road surface conditions for autonomous vehicles," *Journal of Sensors and Sensor Systems*, vol. 11, pp. 187-199, Jun. 2022, doi:https://jsss.copernicus.org/articles/11/187/2022/.

A

Appendix 1

A.1 Hyperparameters

Table A.1: Selected Hyperparameters for all Models. Displaying the configuration space, number of iterations and selected values for the hyperparameters.

ML Model	Hyperparameters	Type	Search Space 1st Iter	Search Space 2nd Iter	Selected
Gaussian NB	var_smoothing	Discrete	[10^{-18}, 10^{-17},..., 10^{-2}, 10^{-1},1]	[0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5]	0.3
	priors	Categorical	[None]	[None]	None
Complement NB	alpha	Discrete	[0, 0.1, 0.5, 1, 5, 10, 100]	[50, 75, 100, 250, 500, 1000]	1000
	force_alpha	Boolean	[True, False]	[True, False]	True
	norm	Boolean	[True, False]	[True, False]	True
	class_prior	Categorical	[None]	[None]	None
HoeffdingTree Classifier	grace_period	Discrete	[50, 100, 200, 300, 350]	[275, 300, 325]	275
	leaf_prediction	Categorical	[‘mc’, ‘nb’, ‘nba’]	[‘mc’, ‘nb’, ‘nba’]	nba
	split_confidence	Discrete	[0.1, 0.25, 0.5, 0.75, 1]	[0.2, 0.25, 0.3]	0.3
	split_criterion	[Discrete]	[‘info gain’]	[‘info gain’]	info gain
Logistic Regression Ensemble	penalty	Categorical	[‘L1’, ‘L2’, ‘elasticnet’] *		elasticnet
	solver	Categorical	[’newton-cg’ ,’sag’, ’saga’,’lbfgs’]		saga
	C	Discrete	[0.01, 0.1, 1.0, 10.0]		10.0
	max_iter	Discrete	[1000]		1000
	L1_ratio	Discrete	[0.5]		0.5
	Ensemble Specific:				
	Vote Type	Categorical	[‘Equal’, ‘Weighted’]		Weighted
	Accuracy Type	Categorical	[‘Regular’, ‘Balanced’]		Regular
	Ensemble Size	Discrete	[1, 5, 10, 15, 20]		15
	Sliding Window	Boolean	[True, False]		True
DecisionTree Ensemble	criterion	Categorical	[‘log_loss’, ‘gini’, ‘entropy’]		log_loss
	max_depth	Discrete	[None, 5, 10, 100]		5
	min_samples_split	Discrete	[2, 8, 14, 20]		14
	min_samples_leaf	Discrete	[0.5, 1, 2, 10]		10
	max_features	Categorical	[None, ‘sqrt’, ‘log2’]		log2
	class_weight	Categorical	[None]		None
	Ensemble Specific:				
	Vote Type	Categorical	[‘Equal’, ‘Weighted’]		Weighted
	Accuracy Type	Categorical	[‘Regular’, ‘Balanced’]		Regular
	Ensemble Size	Discrete	[5, 10, 15, 20]		15
	Sliding Window	Boolean	[True, False]		True
K-Nearest Neighbors Ensemble	n_neighbors	Discrete	[1, 5, 10, 15]		15
	weights	Categorical	[‘uniform’, ‘distance’]		distance
	metrics	Categorical	[‘euclidean’, ‘manhattan’]		euclidean
	Ensemble Specific:				
	Vote Type	Categorical	[‘Equal’, ‘Weighted’]		Weighted
	Accuracy Type	Categorical	[‘Regular’, ‘Balanced’]		Regular
	Ensemble Size	Discrete	[1, 5, 10, 15, 20]		15
	Sliding Window	Boolean	[True, False]		False
Dummy Classifier Ensemble	Strategy	Discrete	[Prior]		Prior
	Ensemble Specific:				
	Vote Type	Categorical	[‘Equal’, ‘Weighted’]		Weighted
	Accuracy Type	Categorical	[‘Regular’, ‘Balanced’]		Regular
	Ensemble Size	Discrete	[1, 5, 10, 15, 20]		20
	Sliding Window	Boolean	[True, False]		False
Single Dummy Classifier	Strategy	Discrete	[Prior]		Prior
Random Forest	estimators	Discrete	[100]		100
	random_state	Categorical	[42]		42
	criterion	Categorical	[‘log_loss’, ‘gini’, ‘entropy’]		log_loss
	max_depth	Discrete	[None, 10, 100]		10
	min_samples_split	Discrete	[2, 8, 14]		14
	min_samples_leaf	Discrete	[0.5, 1, 2]		1
	max_features	Categorical	[‘sqrt’]		sqrt
	class_weight	Categorical	[None]		None

* For tuning LRC: Penalty and solver were combined in specific pairs due to limitations in the provided implementation when running on multiclass problems [12]. The following combinations were used during tuning. [Penalty, Solver]. [None, 'newton-cg'], ['l2', 'newton-cg'], [None, 'sag'], ['l2', 'sag'], [None, 'saga'], ['l2', 'saga'], ['l1', 'saga'], ['elasticnet', 'saga'], ['l2', 'lbfgs'] & [None, 'lbfgs']

A.2 Dataset

Figure A.1: RSC legend. Presenting the colour coding for each RSC and the condition codes used in the data set.

Datapoint examples

Table A.2: Examples of Datapoints from the Dataset. Displaying examples of each RSC with corresponding feature data labelled as X0, X1, X2 and X4. Includes the time of measurement by the laser and the camera capture time (ImgTime), along with coordinates of the laser points in the picture given with ImgX and ImgY.

Time	X0	X1	X2	X3	Condition	ImgTime	ImgX	ImgY
2022-02-24_01-39-55.226812793Z	-1.759391	0.658933	-0.350626	0.325404	6	2022-02-24_01-39-55.226812793Z	1391	1203
2022-02-24_01-42-59.857179531Z	-0.730784	-0.893071	0.043697	-0.503799	5	2022-02-24_01-42-59.857179531Z	1317	1273
2022-02-24_01-40-06.680916285Z	-1.449531	0.641393	-0.122220	-1.277180	4	2022-02-24_01-40-06.680916285Z	1336	1225
2022-02-24_01-39-47.483862923Z	-1.251705	1.509671	-0.774278	-0.751101	2	2022-02-24_01-39-45.948010679Z	1232	1586
2022-02-24_01-39-57.891989682Z	-2.030795	1.682243	-0.762877	1.334712	1	2022-02-24_01-39-55.226812793Z	1355	1203
2022-02-24_01-40-30.139175540Z	-1.316324	0.016451	-0.767736	-0.801484	0	2022-02-24_01-40-27.915000238Z	1303	1200

A.2.1 Assessment Set: The Dataset Used for Training:

Session 1

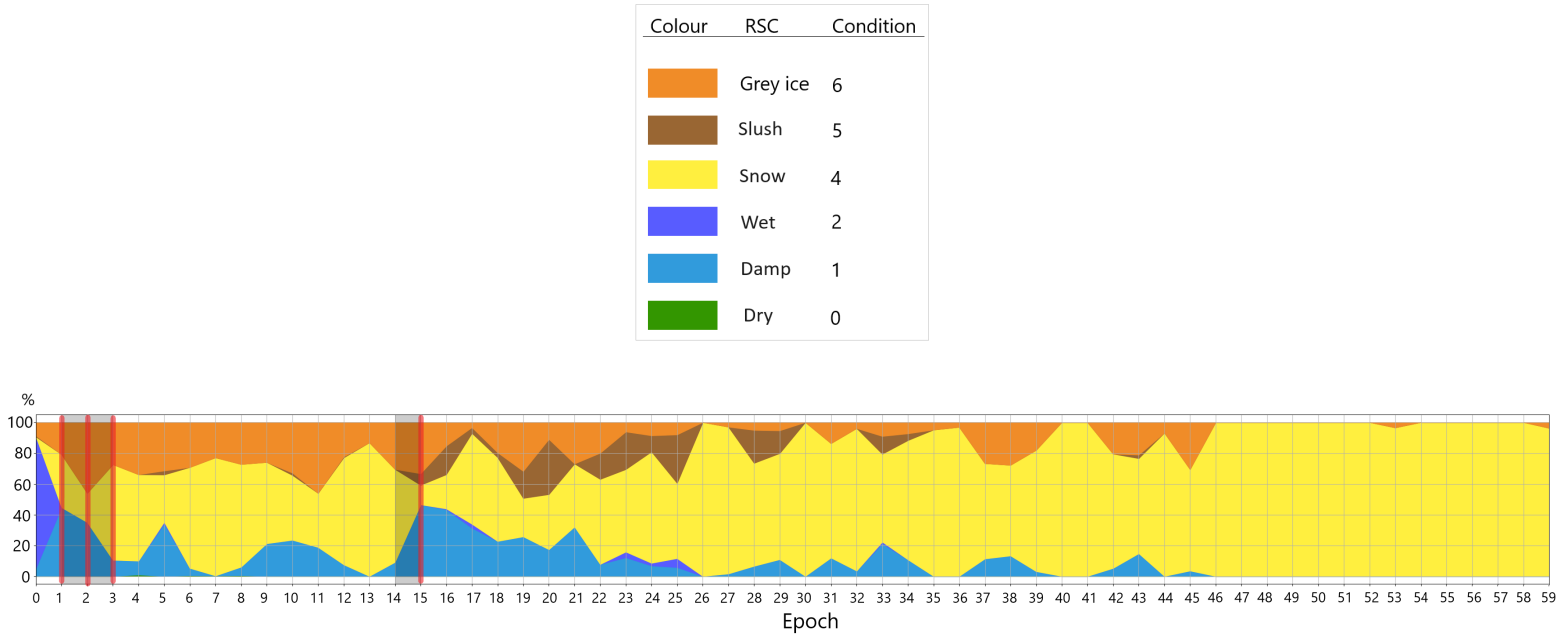


Figure A.2: RSC Distribution in Session 1. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.3: Session 1 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2022-02-24_01-39-40.385856848Z	
Data gathering site	Tokyo	
Length	10 Minutes 599.88 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	299	
Epochs with Concept drift analysis	E1-3 & E14-15	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	18002	14.87
Slush	3828	3.16
Snow	84892	70.12
Wet	2260	1.87
Damp	12029	9.94
Dry	54	0.04
Total data points	121065	100.0

Session 2

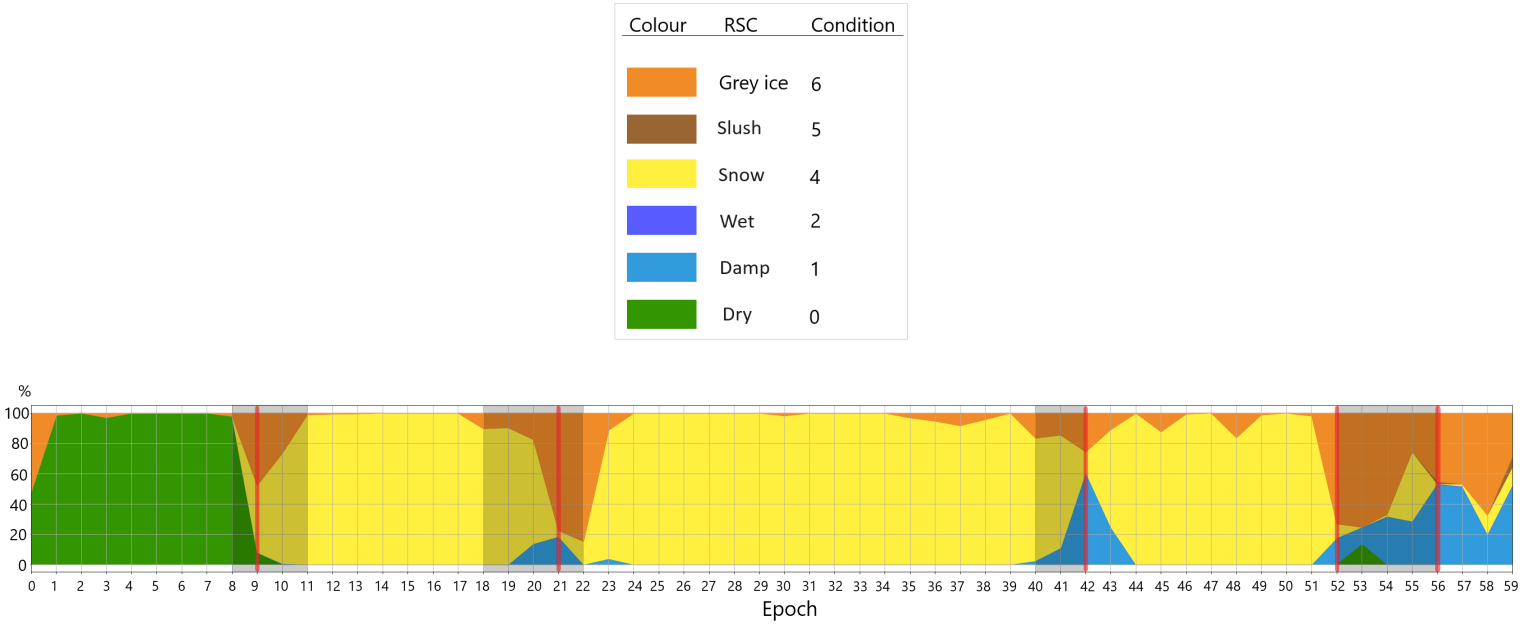


Figure A.3: RSC Distribution in Session 2. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.4: Session 2 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2022-02-24_01-49-47.747852692Z	
Data gathering site	Tokyo	
Length	9.96 Minutes	
	597.90 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	298	
Epochs with Concept drift analysis	E8-11, E18-22, E40-24 & E52-56	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	18999	15.19
Slush	136	0.1
Snow	79222	63.34
Wet	32	0.02
Damp	7941	6.35
Dry	18753	15.00
Total data points	125083	100.0

Session 3

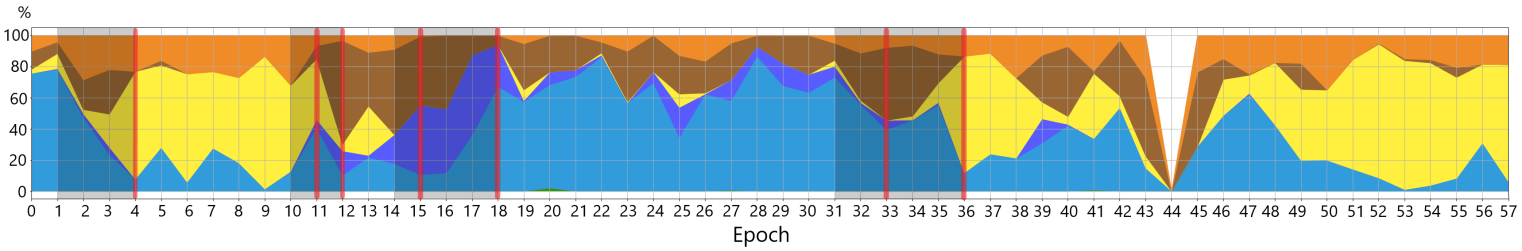
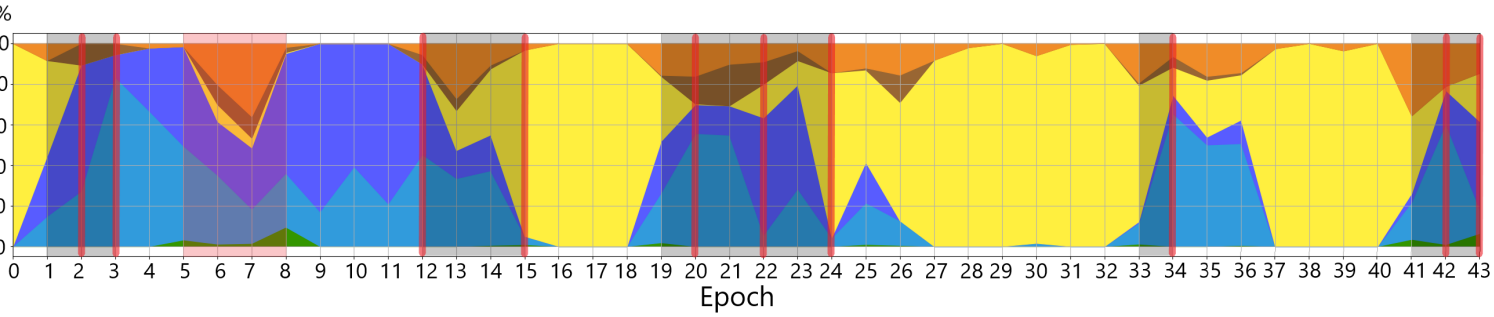


Figure A.4: RSC Distribution in Session 3. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.5: Session 3 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2022-02-24_02-09-47.753886281Z	
Data gathering site	Tokyo	
Length	9.66 Minutes	
	579.82 Seconds	
Number of Epochs (10 s)	57	
Number of Epochs (2 s)	289	
Epochs with Concept drift analysis	E1-4, E10-12, E14-18 & E31-36	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	10610	13.9
Slush	10929	14.3
Snow	23471	30.7
Wet	3247	4.25
Damp	28109	36.8
Dry	40	0.05
Total data points	76406	100.0



Session 4

Figure A.5: RSC Distribution in Session 4. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.6: Session 4 Dataset Summary. Stating detailed information about the session.

Attribute	Value	
Session name	2022-02-25_04-45-56.309832983Z	
Data gathering site	Tokyo	
Length	7.28 Minutes 437 Seconds	
Number of Epochs (10 s)	43	
Number of Epochs (2 s)	218	
Epochs with Concept drift analysis	E1-3, E12-15, E19-24, E33-34 & E41-43	
Analysis of six RSC	E5-8	

RSC Category	Frequency (# of points)	Percentage (%)
Gray Ice	7262	8.62
Slush	2671	3.17
Snow	36801	43.68
Wet	17232	20.45
Damp	19631	23.31
Dry	648	0.77
Total data points	84245	100.0

Session 5

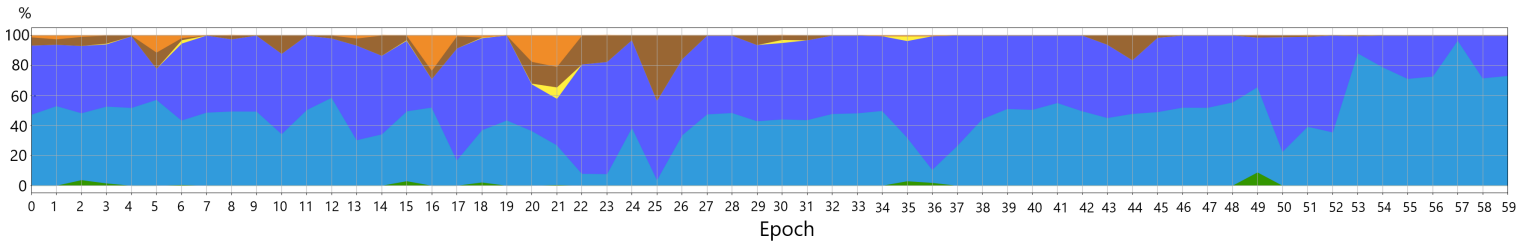


Figure A.6: RSC Distribution in Session 5. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.7: Session 5 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2022-02-25_05-06-02.702329963Z	
Data gathering site	Tokyo	
Length	9.96 Minutes	
	597.83 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	298	
Epochs with Concept drift analysis	-	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	1586	1.23
Slush	5024	3.88
Snow	343	0.27
Wet	61681	47.6
Damp	60472	46.7
Dry	391	0.32
Total data points	129497	100.0

Session 6

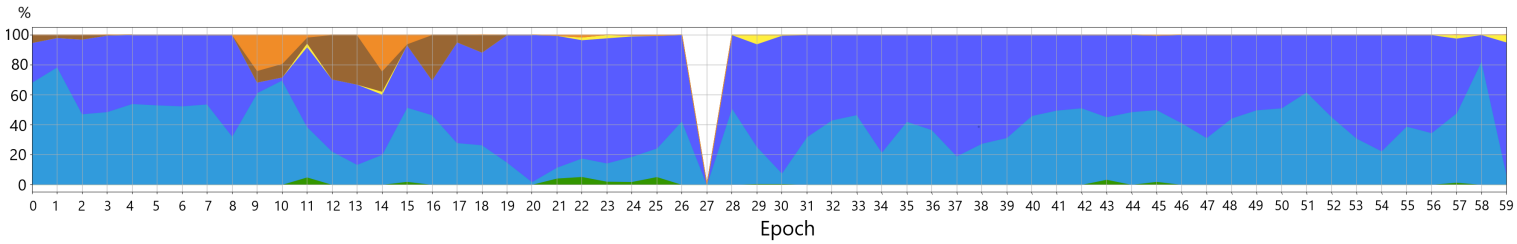


Figure A.7: RSC Distribution in Session 6. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.8: Session 6 Dataset Summary. Stating detailed information about the session, including the RSC distribution in percentage and frequency

Attribute	**Value**	
Session name	2022-02-25_05-16-02.706778792Z	
Data gathering site	Tokyo	
Length	9.97 Minutes	
	597.98 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	298	
Epochs with Concept drift analysis	-	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	2172	1.7
Slush	4137	3.25
Snow	319	0.25
Wet	71919	56.6
Damp	47770	37.6
Dry	759	0.6
Total data points	127076	100.0

Session 7

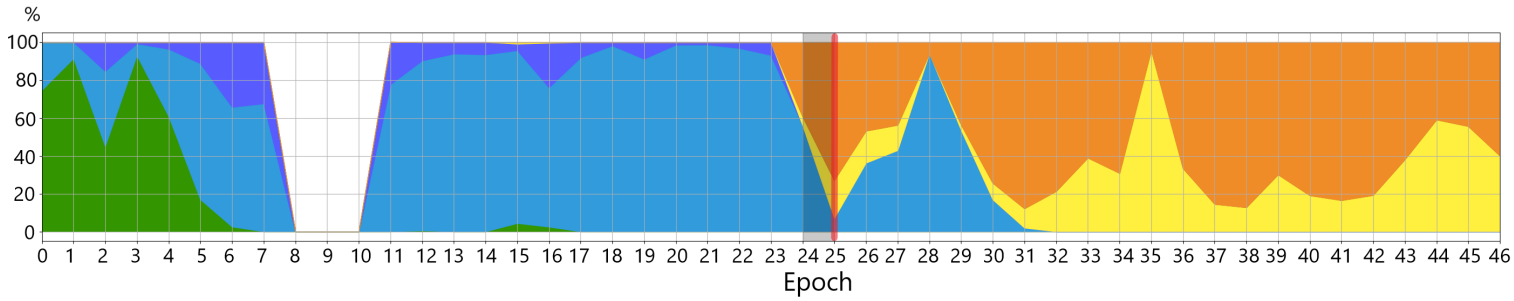


Figure A.8: RSC Distribution in Session 7. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.9: Session 7 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2023-02-16_05-56-04.484101607Z	
Data gathering site	Tokyo	
Length	7.71 Minutes	
	462.5 Seconds	
Number of Epochs (10 s)	46	
Number of Epochs (2 s)	231	
Epochs with Concept drift analysis	E24-25	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	29873	29.8
Slush	0	0.0
Snow	11825	11.8
Wet	3818	3.8
Damp	44729	44.6
Dry	10023	10.0
Total data points	100268	100.0

Session 8

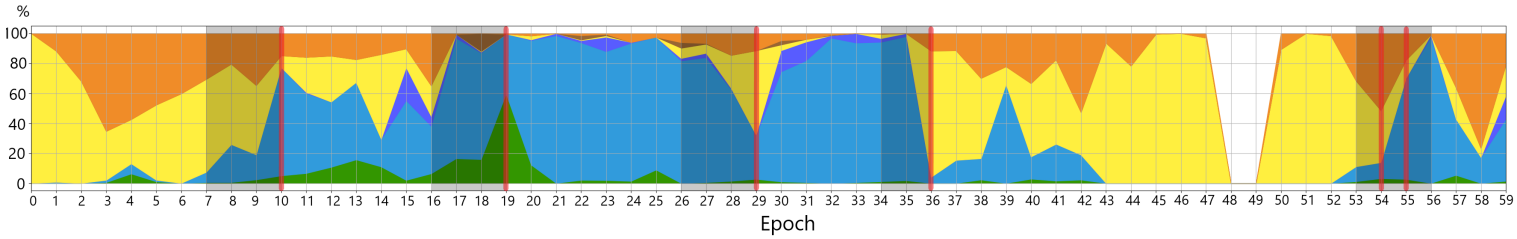


Figure A.9: RSC Distribution in Session 8. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.10: Session 8 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2023-02-16_06-24-16.882150981Z	
Data gathering site	Tokyo	
Length	9.95 Minutes 597.2 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	298	
Epochs with Concept drift analysis	E7-9, E16-19, E26-29, E34-36, & E53-56	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	21376	16.71
Slush	546	0.43
Snow	41984	32.82
Wet	2459	1.92
Damp	56685	44.32
Dry	4861	3.8
Total data points	127911	100.0

Session 9

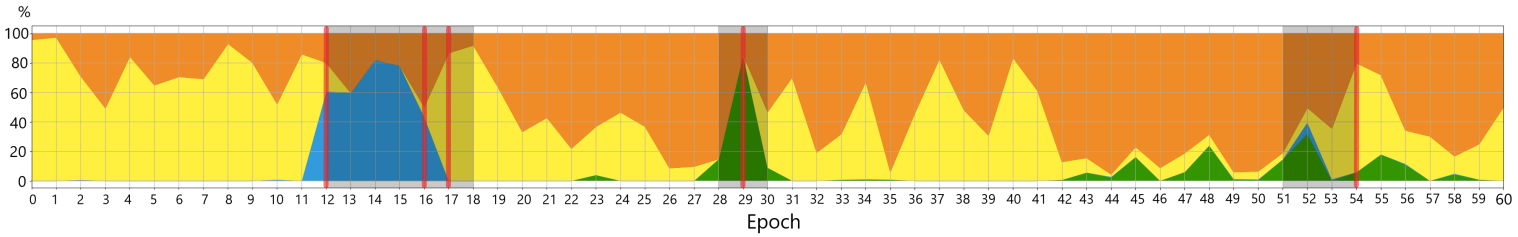


Figure A.10: RSC Distribution in Session 9. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.11: Session 9 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2023-02-16_06-42-30.897661087Z	
Data gathering site	Tokyo	
Length	10.01 Minutes	
	600.82 Seconds	
Number of Epochs (10 s)	60	
Number of Epochs (2 s)	300	
Epochs with Concept drift analysis	E12-18, E28-30 & E51-54	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	71892	50.94
Slush	0	0.0
Snow	52575	37.26
Wet	0	0.0
Damp	9869	7.0
Dry	6781	4.8
Total data points	141117	100.0

Session 10

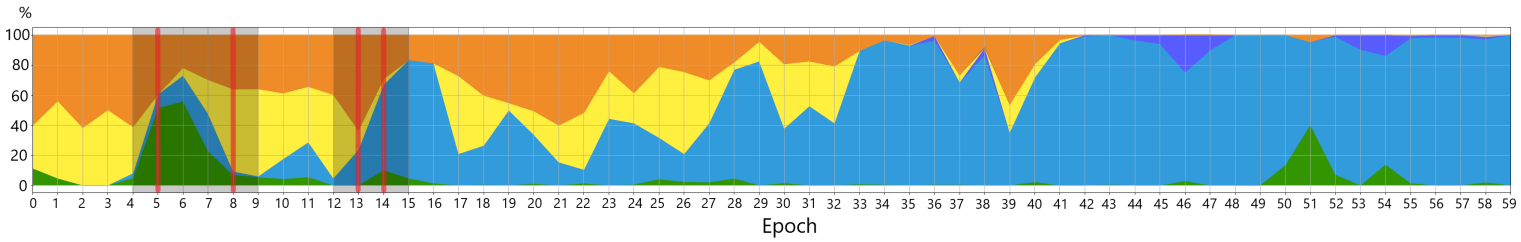


Figure A.11: RSC Distribution in Session 10. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.12: Session 10 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2023-02-16_06-52-39.159042749Z	
Data gathering site	Tokyo	
Length	9.97 Minutes	
	597.98 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	298	
Epochs with Concept drift analysis	E4-9 & E12-15	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	32938	21.85
Slush	24	0.02
Snow	24814	16.46
Wet	2048	1.37
Damp	83300	55.3
Dry	7642	5.0
Total data points	150766	100.0

Session 11

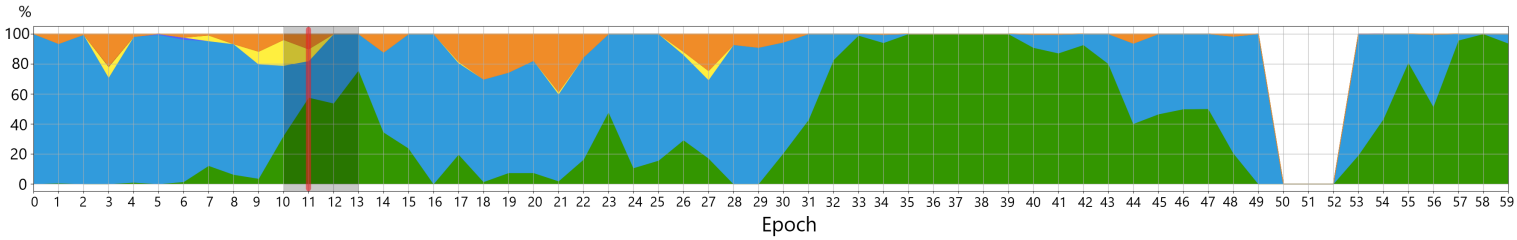


Figure A.12: RSC Distribution in Session 11. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.13: Session 11 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2023-02-16_07-02-39.164508914Z	
Data gathering site	Tokyo	
Length	9.96 Minutes	
	597.6 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	298	
Epochs with Concept drift analysis	E10-13	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	7154	5.61
Slush	9	0.01
Snow	1284	1.0
Wet	54	0.04
Damp	67413	52.87
Dry	51598	40.47
Total data points	127512	100.0

Session 12

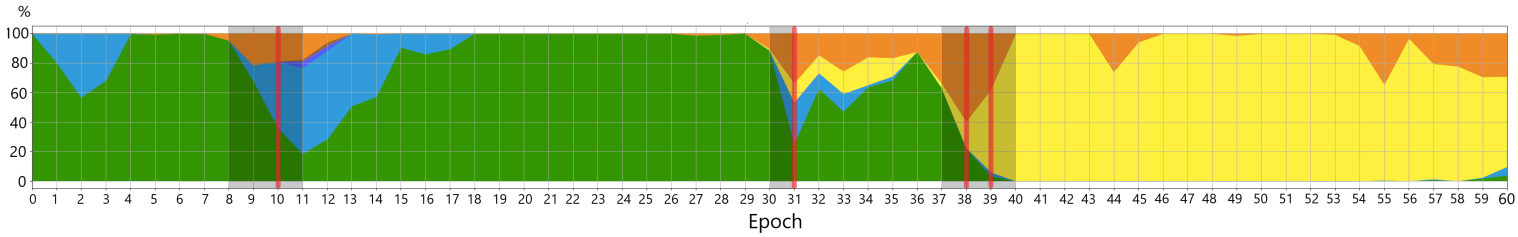


Figure A.13: RSC Distribution in Session 12. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.14: Session 12 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2023-02-16_08-30-44.164517089Z	
Data gathering site	Tokyo	
Length	10.02 Minutes	
	601.38 Seconds	
Number of Epochs (10 s)	60	
Number of Epochs (2 s)	300	
Epochs with Concept drift analysis	E8-11, E30-31 & E37-40	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	11917	8.23
Slush	86	0.06
Snow	44753	30.88
Wet	183	0.13
Damp	11164	7.7
Dry	76780	53.0
Total data points	144883	100.0

Session 13

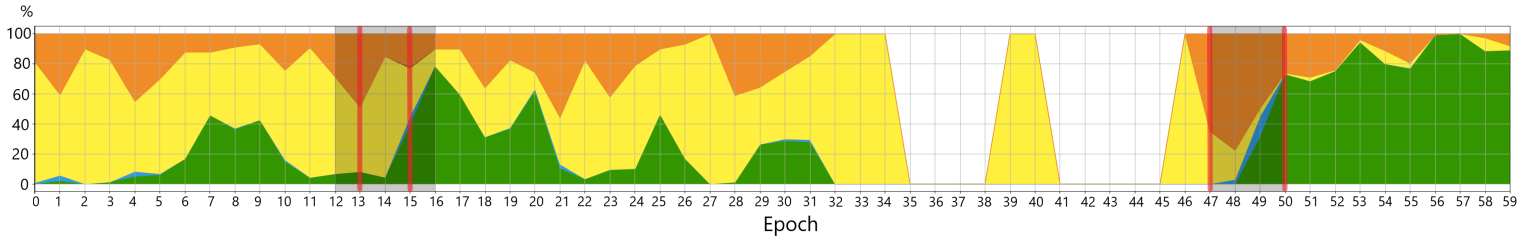


Figure A.14: RSC Distribution in Session 13. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.15: Session 13 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2023-02-16_08-40-50.848336282Z	
Data gathering site	Tokyo	
Length	9.96 Minutes	
	597.8 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	298	
Epochs with Concept drift analysis	E12-16 & E47-50	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	22755	21.45
Slush	32	0.03
Snow	43247	40.78
Wet	0	0.0
Damp	825	0.78
Dry	39216	36.96
Total data points	106075	100.0

Session 14

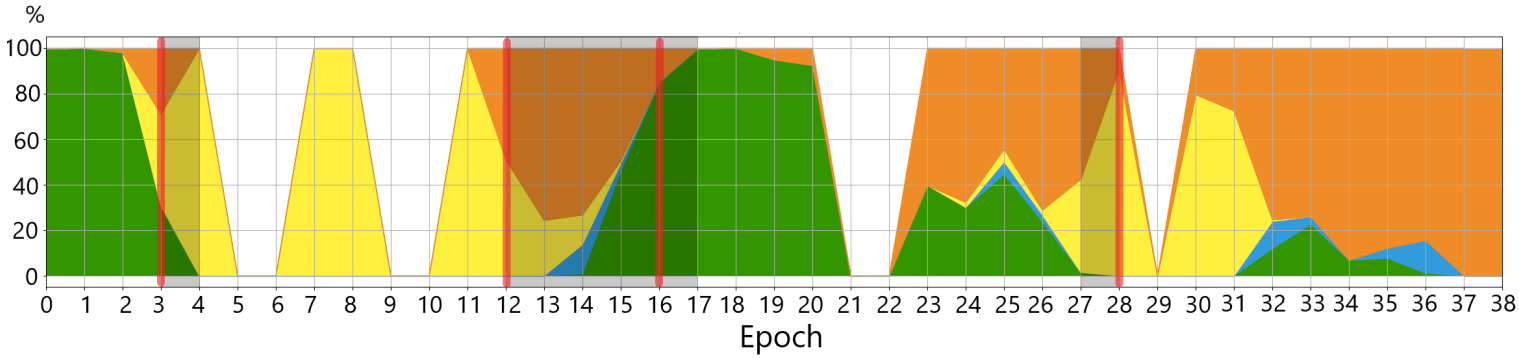


Figure A.15: RSC Distribution in Session 14. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.16: Session 14 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2023-02-16_08-50-50.850470609Z	
Data gathering site	Tokyo	
Length	6.35 Minutes	
	381 Seconds	
Number of Epochs (10 s)	38	
Number of Epochs (2 s)	190	
Epochs with Concept drift analysis	E3-4, E12-17 & E27-28	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	19757	39.77
Slush	0	0.0
Snow	4818	9.7
Wet	0	0.0
Damp	1300	2.62
Dry	23801	47.91
Total data points	49676	100.0

A.2.2 Hyperparameter set: Dataset used for tuning hyperparameters

Session H-1

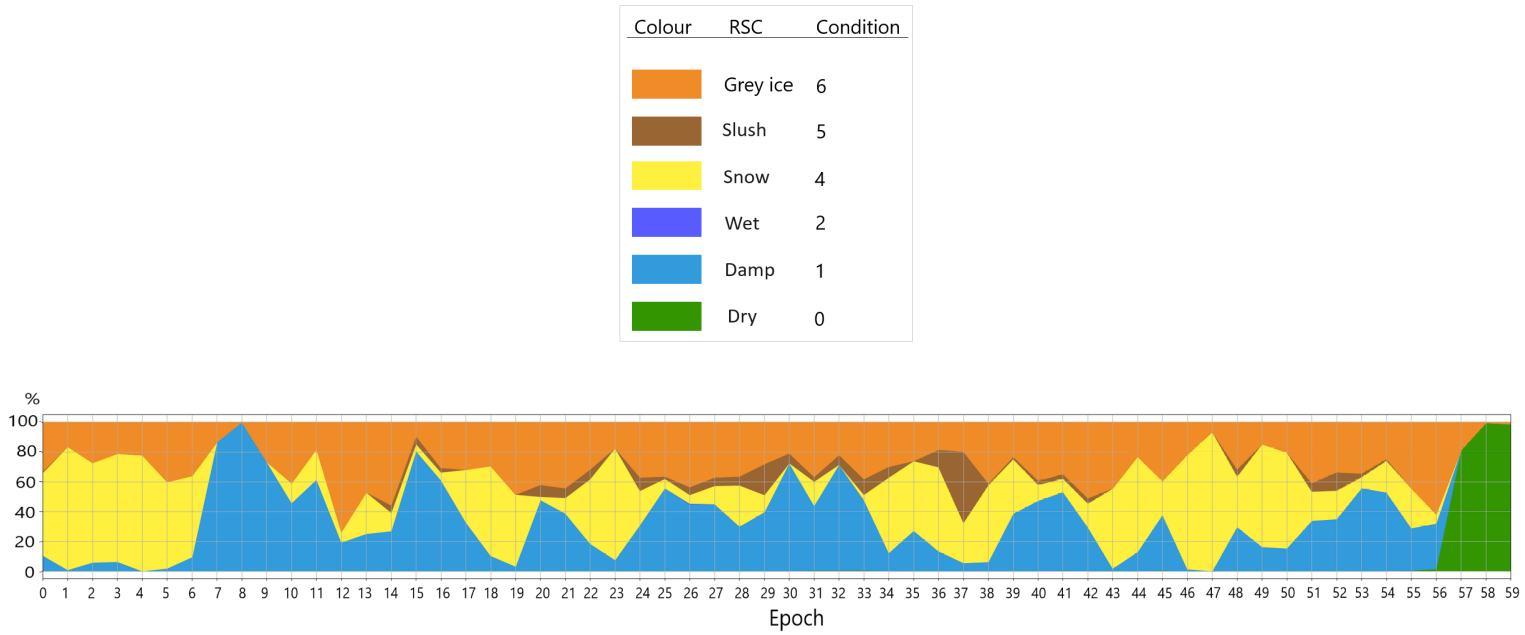


Figure A.16: RSC Distribution in Session H-1. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.17: Session H-1 Dataset Summary. Stating detailed information about the session.

Attribute	Value	
Session name	2022-02-24_01-59-47.749787356Z	
Country	Tokyo	
Length	9.94 Minutes	
	596.5 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	298	

RSC Category	Frequency (# of points)	Percentage (%)
Gray Ice	35477	30.53
Slush	4929	4.24
Snow	34622	29.80
Wet	13	0.01
Damp	36948	31.79
Dry	4233	3.63
Total data points	116222	100.0

Session H-2

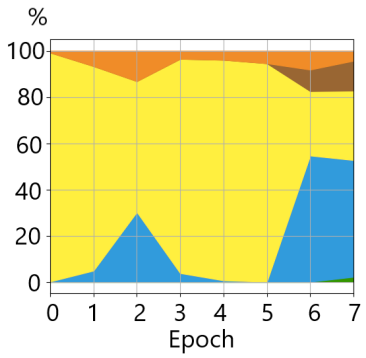


Figure A.17: RSC Distribution in Session H-2. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.18: Session H-2 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2022-02-24_02-19-47.761999931Z	
Country	Tokyo	
Length	1.28 Minutes	
	76.71 Seconds	
Number of Epochs (10 s)	7	
Number of Epochs (2 s)	37	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	395	6.18
Slush	20	0.31
Snow	5382	84.26
Wet	0	0.0
Damp	589	9.22
Dry	2	0.03
Total data points	6388	100.0

Session H-3

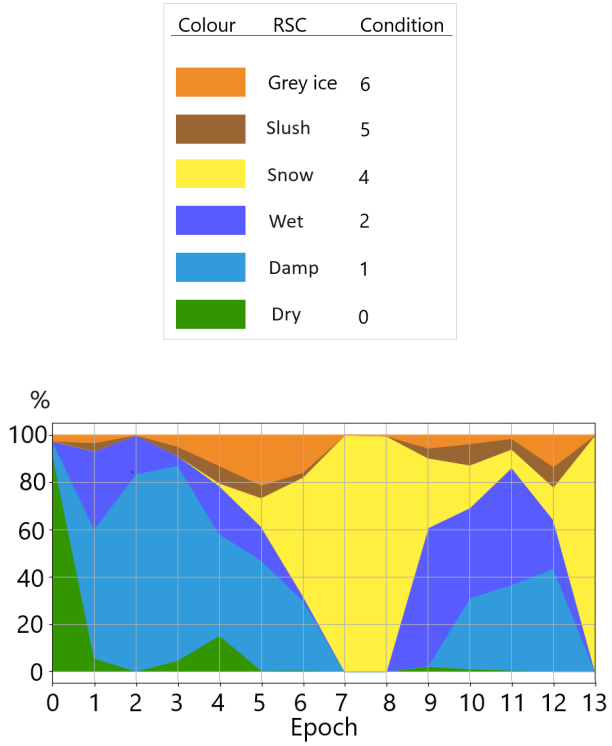


Figure A.18: RSC Distribution in Session H-3. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.19: Session H-3 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2022-02-25_04-31-18.459104739Z	
Country	Tokyo	
Length	2.21 Minutes	
	132.3 Seconds	
Number of Epochs (10 s)	13	
Number of Epochs (2 s)	66	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	1634	5.99
Slush	1015	3.72
Snow	7183	26.32
Wet	5673	20.79
Damp	9788	35.87
Dry	1995	7.31
Total data points	27288	100.0

Session H-4

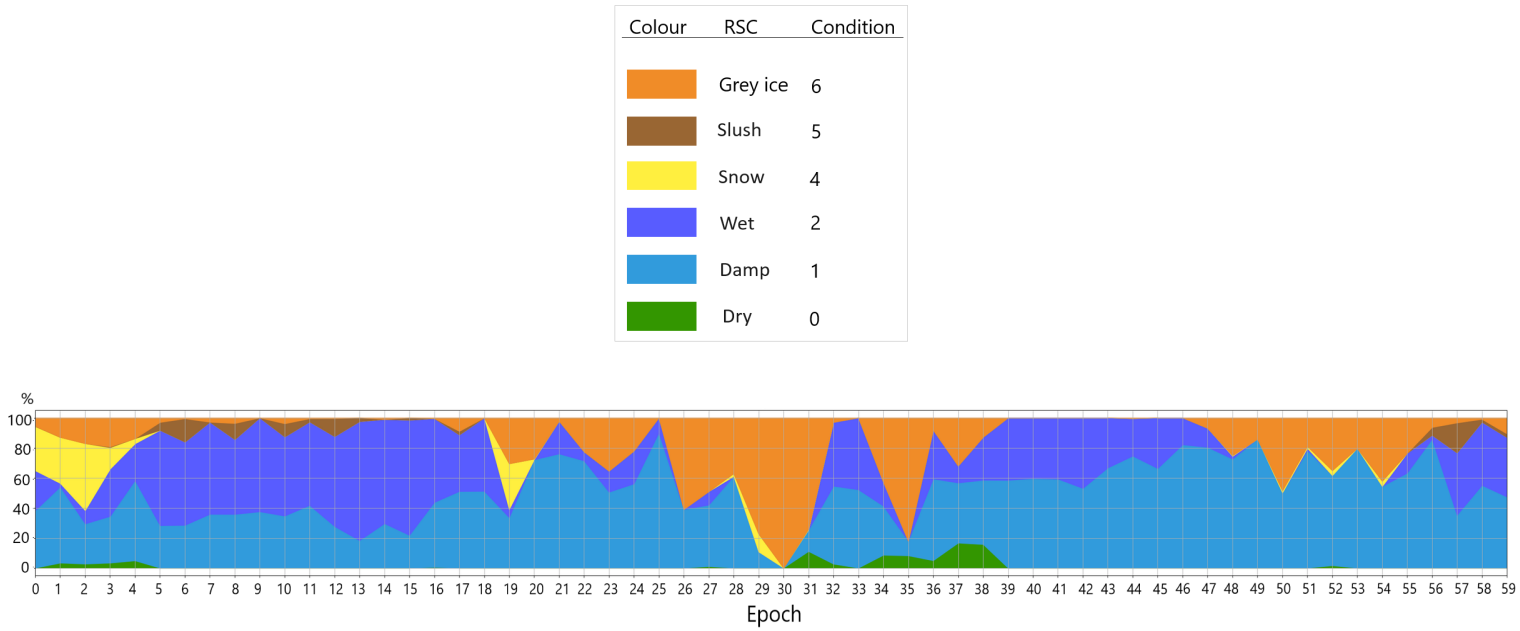


Figure A.19: RSC Distribution in Session H-4. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.20: Session H-4 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2022-02-25_04-56-02.700619979Z	
Country	Tokyo	
Length	9.97 Minutes	
	598 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	299	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	18606	17.7
Slush	1616	1.53
Snow	2655	2.52
Wet	28939	27.5
Damp	51826	49.3
Dry	1517	1.45
Total data points	105159	100.0

Session H-5

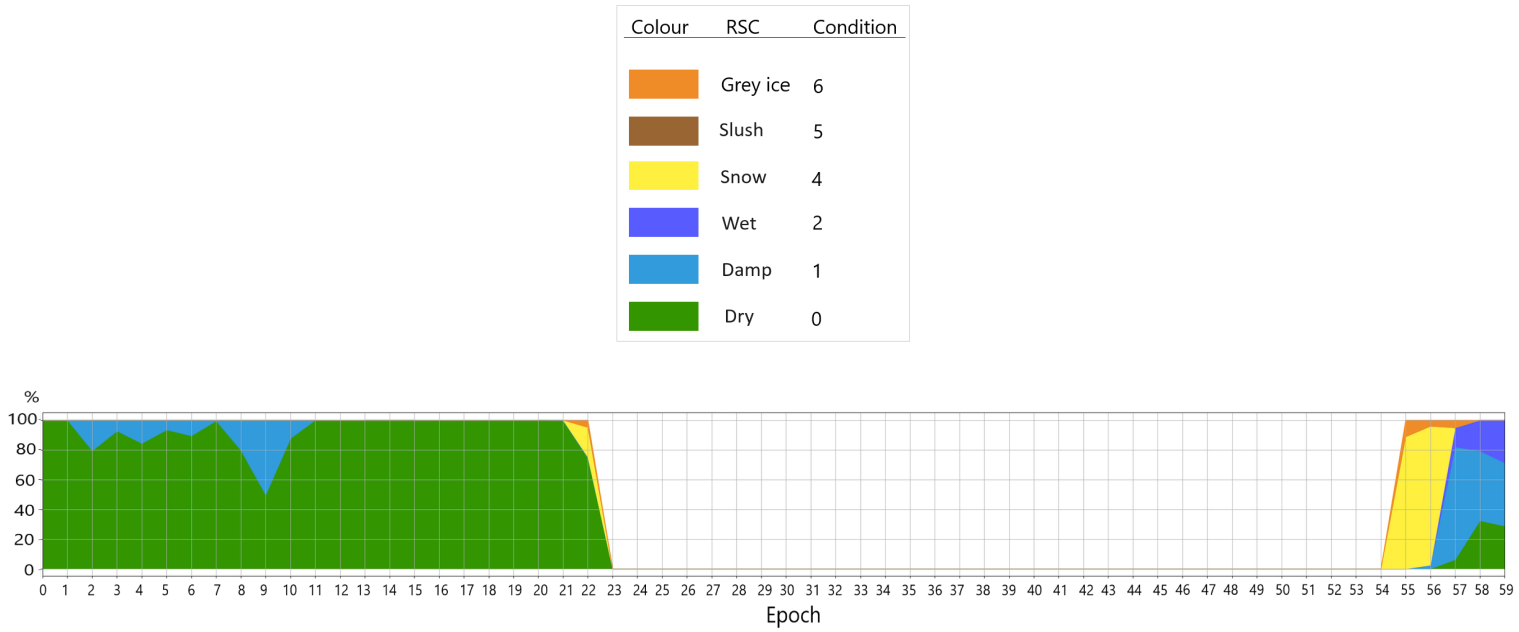


Figure A.20: RSC Distribution in Session H-5. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.21: Session H-5 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2023-02-16_05-46-04.480436678Z	
Country	Tokyo	
Length	9.97 Minutes	
	598 Seconds	
Number of Epochs (10 s)	59	
Number of Epochs (2 s)	299	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	161	0.26
Slush	0	0.0
Snow	1639	2.6
Wet	1403	2.21
Damp	6662	10.51
Dry	53532	84.42
Total data points	63397	100.0

Session H-6

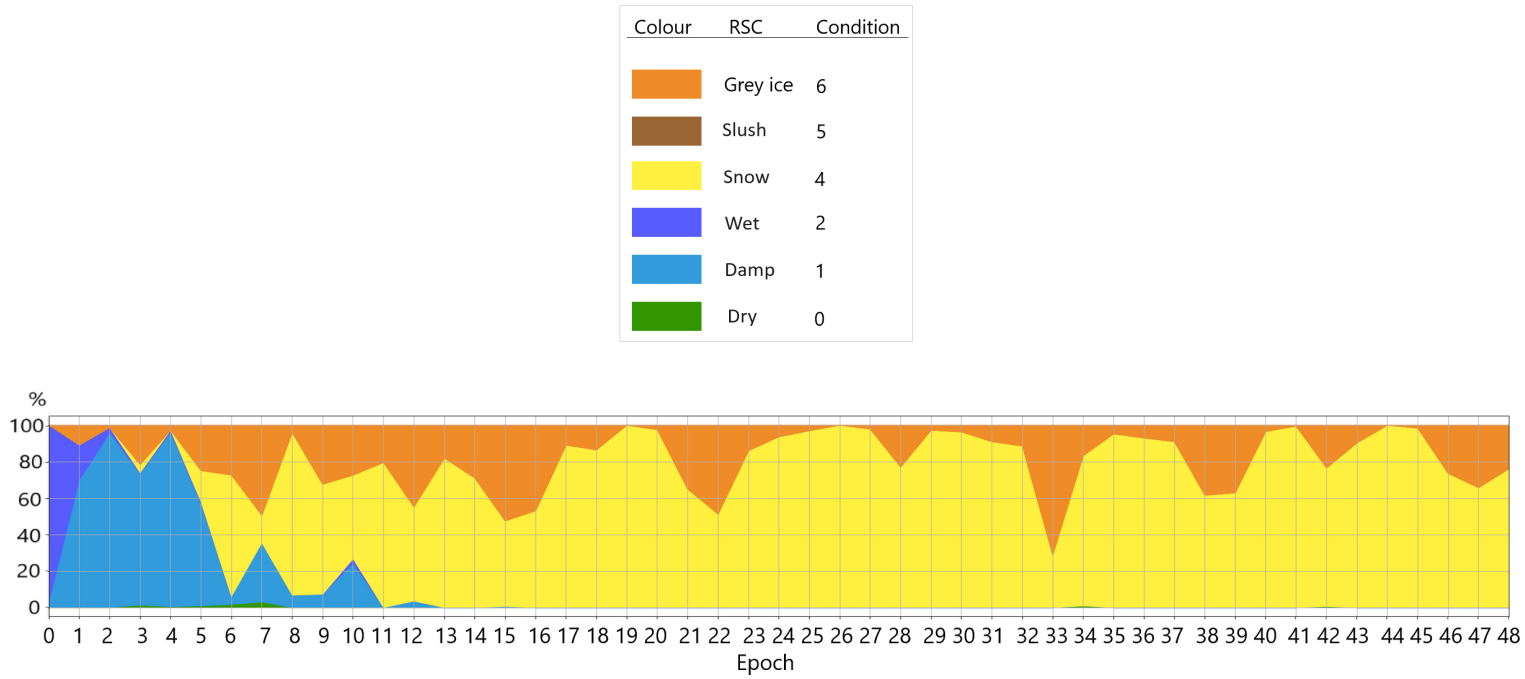


Figure A.21: RSC Distribution in Session H-6. Displaying the RSC distribution over the epochs of the session. Highlighting concept drift-events with vertical red lines and concept drift windows with gray areas.

Table A.22: Session H-6 Dataset Summary. Stating detailed information about the session.

Attribute	**Value**	
Session name	2023-02-16_06-34-16.886430261Z	
Country	Tokyo	
Length	8.12 Minutes	
	487.5 Seconds	
Number of Epochs (10 s)	48	
Number of Epochs (2 s)	243	

RSC Category	**Frequency (# of points)**	**Percentage (%)**
Gray Ice	14650	17.57
Slush	0	0.0
Snow	56804	68.12
Wet	2162	2.6
Damp	9599	11.51
Dry	171	0.2
Total data points	83386	100.0

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY