UNIVERSITY
OF OSLO

Alise Danielle Midtfjord

Machine learning methods for safety-critical systems with time dependency

With applications to airplane landings and environmental data

Thesis submitted for the degree of Philosophiae Doctor

Department of Mathematics
Faculty of Mathematics and Natural Sciences

UNIVERSITAS OSLOENSIS · MDCCCXI ·

2023

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 2615

ISSN 1501-7710

Cover: UiO.
Print production: Graphic Center, University of Oslo.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of *Philosophiae Doctor* at the University of Oslo. The research presented here was conducted at the University of Oslo, under the supervision of professor Arne Bang Huseby and associate professor Riccardo De Bin. This work was supported by the Faculty of Mathematics and Natural Sciences.

The thesis is a collection of four papers, which explores challenges related to the use of machine learning methods for safety-critical systems, and addresses these problems by development of methods, algorithms and novel machine learning models. The papers are presented in chronological order of writing, and is preceded by an introductory chapter that relates them to each other and provides background information and motivation for the work.

Acknowledgements

I would like to thank my supervisors Arne Bang Huseby and Riccardo De Bin for their invaluable contributions during the work with this thesis. I am very grateful that you gave me freedom to explore my ideas, while still being available for advice and encouragement. In particular, I want to thank my main supervisor and main collaborator Arne for the cooperation during our three joint papers. You have generously shared of your time to provide me with data and relevant knowledge. I appreciate that your door was always open for all questions as well as nice conversations, and for the pleasant travels we had to conferences and research trips together. I would also like to thank Riccardo especially for sharing your knowledge and passion, which has greatly enhanced my understanding of the field. I am also grateful to Avinor for making data available for this research project.

Additionally, I want to thank Mari Dahl Eggen, for our close collaboration during the last paper. We faced several challenges trying to combine two fields into a novel methodology, but I enjoyed all parts of our journey together. Thank you for being patient in the more stressful times, and for always being available for discussions.

Finally, I want to thank my friends and family for their support during these years. Especially, I want to thank Helene B. Midtfjord for inspiring me into pursuing the Ph.D., as well as Simen Wilsher-Lohre for your constant support and patience when I had to prioritize the research work.

Alise Danielle Midtfjord
Oslo, May 2023

Sammendrag

Kunstig intelligens og maskinlæring har en økende bruk i mange bransjer og applikasjoner, der stadig flere og viktigere beslutninger blir tatt på bakgrunn av maskinlæringsmodeller. Riktignok har nyere forskning synliggjort flere begrensninger ved mange av de mest brukte maskinlæringsmetodene, eksempelvis utfordringer knyttet til stabilitet, transparens og manglende informasjon om usikkerheten til modellenes prediksjoner. Ettersom maskinlæringsmodeller begynner å få innvirkning på mange områder av menneskers liv, er det avgjørende å sikre trygg og ansvarlig bruk av kunstig intelligens og maskinlæring. Spesielt innenfor sikkerhetskritiske felt som helse, transport, risikoanalyse og autonome systemer er det viktig å ha pålitelige modeller. Her kan feil beslutninger gi katastrofale utfall som alvorlig skade på utstyr, miljø og mennesker, eller til og med tap av menneskeliv.

Denne avhandlingen tar for seg noen av utfordringene knyttet til bruken av maskinlæring for sikkerhetskritiske systemer, både når det gjelder utforskning samt utvikling av nye modeller, teori og algoritmer. Sikkerhetskritiske systemer innebærer ofte komplekse fysiske fenomener med stor grad av usikkerhet. Dette krever modeller som tillater stor kompleksitet, et område hvor maskinlæring ofte viser overlegenhet ovenfor andre modelleringsmetoder, også tidvis ovenfor menneskelig kunnskap og erfaring. Bruken av avanserte statistiske metoder som maskinlæring kan dermed bidra til økt forståelse og redusering av risiko i slike systemer, hvis de blir anvendt på riktig måte. I tillegg til kompleksitet involverer mange sikkerhetskritiske systemer en form for tidsavhengighet og minne, som betyr at den nåværende tilstanden til systemet er avhengig av tiden og systemets tidligere tilstander. Dermed involverer analysen av slike dynamiske systemer tidsrekker av data, der avhengigheten mellom datapunkter på tvers av tid må tas med i beregningen.

Avhandlingen utforsker spesielt metoder knyttet til to anvendelsesområder, begge relatert til modellering av tidsavhengige miljødata. Det første anvendelsesområdet er relatert til å skape tryggere flylandinger på vinterstid basert på maskinlæring. I områder med kaldt klima kan kontaminering av rullebaner med materialer som snø, is og sludd føre til vanskelige landingsforhold, ettersom den tilgjengelige friksjon mellom flydekk og rullebaneoverflate blir betydelig redusert, som i verstefall kan føre til ulykker. Innenfor denne anvendelsen utforsker avhandlingen problemer relatert til utnyttelsen av sensormålinger fra landende fly til å beregne tilgjengelig friksjon på rullebanen. Vi viser blant annet effekten på beregnet friksjon av å kalibrere ustabile sensormålinger av flyets akselerasjon. Dette påvirker igjen forskjellige anvendelser av den beregnede friksjonen, slik som evaluering av rullebanerapporter og trening av maskinlæringsmodeller. Ved bruk av den beregnede friksjonen utvikler vi maskinlæringsmodeller for å gi prognoser

av tilgjengelig friksjon basert på en gradient boosting algoritme på tidforsinkede miljødata som temperatur, nedbør og fuktighet. Disse modellene kombineres med forklarlig kunstig intelligens, altså metoder som gjør at mennesker kan forstå begrunnelsene for avgjørelsen tatt av maskinlæringsmodeller. Vi illustrerer hvordan kombinasjonen av presise maskinlæringsmodeller og forklarlig kunstig intelligens kan skape et tillitsverdig beslutningsstøttesystem for tryggere flylandinger.

En annen utfordring relatert til beregning av friksjonen på rullebaner er manglende verdier for mesteparten av flylandingene grunnet begrensninger i beregningsmetoden. For disse landingene har vi kun beregninger av en nedre grense for den sanne friksjonen. Inspirert av dette problemet, utvikler vi en ny metode for håndtering av venstresensurert data, det vil si data der kun en nedre grense for den sanne verdien er observert. Denne metoden, Clayton-boost, er en gradient boosting algoritme basert på en akselerert feiltid modell (AFT). Flydataene synliggjør at den sanne verdien kan være høyt avhengig av sensureringsmekanismen, noe som bryter med betingelsene for de fleste statistiske metoder som AFT modeller. Derfor benytter Clayton-boost seg av en Clayton kopula, for å modellere avhengigheten mellom den sanne verdien og sensureringsmekanismen. Analyser viser at modellen er svært effektiv på å predikere friksjonen selv med en stor andel sensurert data, samt at modellen er effektiv på andre datasett med avhengig venstresensurering.

Det andre anvendelsesområdet er relatert til modellering av klima og ekstremvær. Jordens miljø er et komplekst, ulineært dynamisk system med mye usikkerhet, noe som begrenser tiden fremover man kan lage presise værprognoser. For å få bedre modeller for været på jordoverflaten, kan man imidlertid inkludere faktorer fra stratosfæren, altså det atmosfæriske lageret som ligger rundt 15-50 km over jordoverflaten. Spesielt et ekstremværfenomen kan forstås bedre ved å se på stratosfæren; brå stratosfæriske oppvarminger oppstår når retningen på polarvinden i stratosfæren plutselig snur på vinterstid, som fører til en brå oppvarming av stratosfæren. En brå oppvarming av stratosfæren øker sjansen for ekstremvær på jordoverflaten. For å bedre forutsi sjansene for slikt ekstremvær, er det nyttig å lage prognosemodeller for polarvinden i stratosfæren.

Inspirert av utfordringene knyttet til det stratosfæriske været, utvikler vi en ny maskinlæringsmetode kalt Delay-SDE-net. Dette er en nevral nettverksmodell basert på stokastiske differensiallikninger med forsinkelse (SDDE), som gjør den til en passende modell for tidsserier med avhengighet bakover i tid, ettersom modellen inkluderer tidligere tilstander av systemet for å forutsi fremtidige tilstander. Den stokastiske delen av Delay-SDE-net gir et rammeverk for å estimere usikkerheten i modellering. Vi viser at modellen er svært godt egnet for stratosfærisk data, i tillegg til generelle dynamiske modeller som kan representeres som stokastiske differensiallikninger med forsinkelse.

Målet med utviklingen av nye metoder inspisert av de to anvendelsesområdene er å bidra til å gjøre det tryggere å anvende beslutninger basert på maskinlæring med fokus på sikkerhetskritiske anvendelser med tidsavhengighet.

Abstract

Artificial intelligence and machine learning are increasingly used in many industries and applications, where more and more important decisions are being made based on machine learning models. However, recent research has shown several limitations to many of the commonly used machine learning methods, such as challenges related to stability, transparency and uncertainty. As machine learning models are increasing their impact on many areas of people's lives, it is critical to ensure safe and responsible use of artificial intelligence and machine learning. Especially in safety-critical areas such as health, transport, risk analysis and autonomous systems, it is important to have trustworthy models. In these areas, wrong decisions can lead to catastrophic consequences such as serious harm to equipment, the environment, and people, or even loss of lives.

This thesis addresses some of the challenges related to the use of machine learning for safety-critical systems, both in terms of exploration and development of new models, theory and algorithms. Safety-critical systems often involve complex physical phenomena with a large degree of uncertainty. This requires models that handles high complexity, an area where machine learning often show superiority over other modelling methods, sometimes also over human knowledge and experience. The use of advanced statistical methods such as machine learning can thus contribute to an increased understanding and reduction of the risk associated with such systems if they are used in a proper way. In addition to complexity, many safety-critical systems involve some form of time dependency and memory, meaning that the current state of the system is dependent on time and the system's previous states. Thus, the analysis of such dynamic systems involves time series of data, where the dependency between data points across time must be considered.

The thesis particularly explores methods connected to two applications, both related to modelling of time-dependent environmental data. The first area of application is connected to creating safer airplane landings in winter time based on machine learning. In areas with cold climates, contamination of runways with materials such as snow, ice and sleet can lead to difficult landing conditions, as the available friction between the tires and the runway surface is significantly reduced, which in the worst case can lead to major accidents. Within this application, the thesis explores problems related to the utilization of sensor measurements from landing airplanes to calculate the available friction on the runway. Among other things, we show the effect of calibrating unstable sensor measurements of the acceleration of the landing aircraft on the calculated friction. This in turn affects various applications of the calculated friction, such as evaluation of runway reports and training of machine learning models. Based on the calculated friction, we develop machine learning models to provide

predictions of available friction based on a gradient boosting algorithm on time-delayed environmental data such as temperature, precipitation and humidity. These models are combined with the use of explainable artificial intelligence, i.e., methods that enable humans to understand the reasons for the decision made by machine learning models. We illustrate how the combination of precise machine learning models and explainable artificial intelligence can create a trustworthy decision support system for safer airplane landings.

Another challenge related to calculating the friction on runways is that majority of airplane landings have missing values due to limitations in the friction calculation method. For these landings, we only have calculations of a lower bound for the true friction. Inspired by this problem, we develop a new method for handling left-censored data, i.e., data where only a lower bound of the true value is observed. This method, Clayton-boost, is a gradient boosting algorithm built upon an accelerated failure time model (AFT). The flight data makes it clear that the true value can be highly dependent on the censoring mechanism, which violates the conditions for most statistical methods such as AFT models. Therefore, Clayton-boost makes use of a Clayton copula to model the dependence between the true value and the censoring mechanism. Analyses show that the model is very effective at predicting the friction even with a high percentage of censoring, and that the model is effective on other data sets with dependent left censoring.

The second area of application is related to the modeling of climate and extreme weather. The Earth's environment is a complex, non-linear dynamical system with a lot of uncertainty, which limits the time ahead in which precise weather forecasts can be made. However, to achieve better models for the weather on the earth's surface, one can include factors from the stratosphere, i.e., the atmospheric layer that lies around 15-50 km above the earth's surface. Particularly one extreme weather phenomenon can be better understood by involving factors from the stratosphere. Sudden stratospheric warming occurs when the direction of the polar wind in the stratosphere suddenly reverses in winter, which leads to a sudden warming of the stratosphere. A sudden warming of the stratosphere increases the chance of extreme weather on the earth's surface. To better predict the chances of such extreme weather, it is useful to create forecast models for the polar wind in the stratosphere.

Inspired by the challenges related to stratospheric weather, we are developing a new machine learning method called Delay-SDE-net. This is a neural network model based on stochastic delay differential equations (SDDE), which makes it a suitable model for time series with memory, as the model includes past states of the system to predict future states. The stochastic part of the Delay-SDE net provides a framework for estimating the uncertainty in modelling. We show that the model is very well suited for stratospheric data, in addition to general dynamical models that can be represented as stochastic differential equations with delay.

The aim of the development of new methods inspired by the two application areas is to contribute to safer decision-making based on machine learning, with a special focus on safety-critical applications with time dependence.

List of Papers

Paper I

Midtfjord, A. D. and Huseby, A. B. “Estimating Runway Friction Using Flight Data”. In: *Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference.* (2020), DOI: 10.3850/978-981-14-8593-0.

Paper II

Midtfjord, A. D., De Bin, R. and Huseby, A. B. “A decision support system for safer airplane landings: Predicting runway conditions using XGBoost and explainable AI”. In: *Cold Regions Science and Technology.* Vol. 199, no. 103556 (2022), DOI: 10.1016/j.coldregions.2022.103556.

Paper III

Midtfjord, A. D., De Bin, R. and Huseby, A. B. “A copula-based boosting approach for time-to-event prediction”. Submitted for publication.

Paper IV

Midtfjord, A. D., Eggen, M. D. “Delay-SDE-net: A deep learning approach for time series modelling with memory and uncertainty estimates”. Submitted for publication.

Contents

Chapter 1

Introduction

Artificial intelligence (AI) and machine learning have in the recent years attracted high attention due to their prediction accuracy, limited need for defining parametric distributions, and ability to model complex relations. In a lot of different areas, machine learning models are becoming substitutions for the commonly used models such as statistical, numerical, mathematical as well as engineering and physics models. In some areas, they also substitute human expertise.

However, in the recent years, a lot of applications and analysis has exposed limitations to the frequently used machine learning models, such as instability and lack of uncertainty estimates and interpretability, which all decrease the trustworthiness of the machine learning predictions. In many application areas, it is highly important to know when to trust a prediction and when not to, especially for safety-critical applications such as autonomous systems, risk analysis, environmental predictions and medicine.

This thesis addresses some of the challenges with the use of machine learning for safety-critical problems, both in terms of exploration and development of novel models, theory and algorithms. We explore two motivational cases related to modelling of time-dependent environmental variables, namely airplane landings on slippery runways and sudden stratospheric warmings. These cases shed light on challenges related to modelling complex inference in a high risk system where there is not much room for uncertainty and wrong predictions. The explorative work in Paper I presents challenges related to data sources for the assessment of slippery runways, such as calibration of unreliable sensor measurements and lack of data covering all situations. Paper II addresses some challenges with predicting the runway surface friction by combining existing machine learning and artificial intelligence methods, however tailoring the methods to handle time series data with missing values. Inspired by the challenges in the two first papers, two novel machine learning models are proposed to address some typical challenges which arises when working with safety-critical applications within the time domain. Specifically, Paper III introduces Clayton-boost, a novel boosting model that handles dependent right censoring, which is a type of missing values where one only observes a lower bound of the true value. Paper IV was additionally inspired by challenges related to predicting the occurrence of sudden stratospheric warmings, to provide better forecasts of extreme weather. This paper introduces the Delay-SDE-net, a neural network algorithm which is suitable for time series prediction which also provides estimates of the model uncertainty.

Python scripts were developed as part of all papers, and the code and algorithms for the papers are published at https://github.com/alimid.

In the rest of the introduction, we present some background theory for the papers in Chapter 1.1 and 1.2, as well as context in Chapter 1.3 and Chapter 1.4. Summary of the main contributions of each paper is given in Chapter 1.5.

1.1 Machine Learning Models

The field of statistics is constantly challenged by the problems and tasks originating in other areas of science and application areas. In the recent years, these problems have transformed from statistical inference and linear models to more complex problems with large amounts of data. Machine learning has in the newer years enabled a new type of model creation to suite these problems, as it gives computers the ability to *learn* instead of being explicitly programmed (Faria 2018).

1.1.1 Statistical learning

Let $X \in \mathbb{R}^d$ be the input variables and $Y \in \mathbb{R}$ the output variable with joint probability distribution $\Pr(X, Y)$. When training a statistical model, one is provided a training set of data made of n observations, which is given by

$$D = (\mathbf{x_1}, y_1), \cdots, (\mathbf{x_n}, y_n). \tag{1.1}$$

The task of statistical learning is to find a function $f : \mathbb{R}^d \to \mathbb{R}$ such that $y = f(\mathbf{x}) + \epsilon$, where $f(\mathbf{x})$ is a good prediction for y, $\hat{y} = f(\mathbf{x})$, with an error ϵ. A learning algorithm tries to minimize the expected prediction error ϵ by finding the optimal f within the possible class of functions $\mathbb{H}$, called the *hypothesis space*, given the training data D. This task requires a loss function $L(y, f(\mathbf{x}))$ which penalizes prediction errors. The error ϵ can origin from several sources of uncertainty, which are often categorized as either *aleatoric* or *epistemic* uncertainty. Aleatoric uncertainty refers to natural randomness inherent in a specific task, which is considered impossible to reduce. Epistemic uncertainty arise due to lack of knowledge within a model, and can be reduced by finding a more optimal prediction model f. More details about uncertainty is given in Section 1.4.1

There are many different possible functions for f, algorithms to find the optimal f and variations of loss functions $L(y, f(\mathbf{x}))$, ranging from simple linear regression to complex deep neural networks. The choice of model function and loss function depends on the problem to solve, where the papers in this work address regression, classification, time-to-event and time series prediction. In addition, the models must take into account the bias-variance trade-off. When a model is fitted to the training data, $\hat{f}(x; D)$, the model bias is the errors in defining the wrong model for the underlying function,

$$\text{Bias}(\hat{f}(x; D)) = E[\hat{f}(x; D)] - f(x), \tag{1.2}$$

which is high if the model is not complex enough to capture the underlying function, meaning the hypothesis space of the model does not contain the optimal function f. Model bias can also increase if the model is *underfitted* to the training data, meaning the model is not trained well enough within its hypothesis space due to e.g. too few data point or training iterations. Models with errors originating from model limitations is referred to having epistemic uncertainty.

The model variance is the variability of the model predictions around its expected value,

$$\text{Var}(\hat{f}(x;D)) = E[(E(\hat{f}(x;D)) - \hat{f}(x;D))^2], \tag{1.3}$$

which is high if the model is *overfitted* to the training data, which means it does not generalize well to unseen data. To achieve a trained model with good bias-variance balance, model parameters can be tuned by the use of cross validation or validation datasets, and by choosing models with appropriate complexity.

The language of statistical learning is quite broad where different terms are used quite interchangeably. For example, the input variable X can also be referred to as *covariates*, *explanatory variables*, *features* or simply *variables.* The output variable Y can also be referred to as the *response variable* or simply *response.* Two terms that are quite mixed together are statistics and machine learning. However, while statistics has a main focus on *inference*, which is often achieved through fitting task-specific, and often parametric, probability models, *machine learning* focuses on *prediction* by using generalized learning algorithms to find patterns in often complex and large amounts of data. (Bzdok, Altman, and Krzywinski 2018). Nevertheless, many of the methods from statistics and machine learning can be used for both prediction and inference tasks. The term machine learning is often also intertwined with AI, whereas this thesis refers to AI as any system that mimic the intelligent behaviour of humans. Machine learning, on the other hand, is referred to as the methods of AI which enables machines to learn from data. We will provide some background information about the the machine learning models used in this thesis, which are *gradient boosting* and *neural networks.*

1.1.2 Gradient boosting

Boosting models are some of the most powerful learning models in modern time, as they are very efficient in both lowering bias as well as variance (Breiman 1996). The idea behind boosting started with a theoretical question (Kearns and Valiant 1994), can a set of weak learners become a strong learner? A weak learner is a model which only slightly connects the covariates with the true response, such as a simple linear model, or a simple decision tree. This laid ground for the concept of boosting (Mayr et al. 2014), namely that a weak learner can iteratively be improved to become a strong learner, which lead to the development of the first boosting models in Schapire 1990 and Freund 1995. However, the boosting models do not repetitively apply the same weak learner to the same input data, as this would not alter the prediction from time to time. Instead, the concept of boosting is to apply the weak learner to different data, by iteratively re-weighing the observations based on the performance of the previous iterations, such that each iteration provides a new model $f(x)_k$, and the final model is an ensemble of all the iterations,

$$\hat{f}(x) = \sum_{k=1}^{K} f(x)_k. \tag{1.4}$$

The re-weighting of the observations are done such that observations which got misclassified at the previous iteration gets more attention in the next, making the model focus on the most interesting data structures and better explore the hypothesis space.

The algorithm that made the boosting concept gain much attention is *AdaBoost* by Freund, Schapire, et al. 1996, a still popular boosting algorithm. AdaBoost was the first adaptive boosting algorithm, meaning it automatically updates the observation weights, as well as the contributing weight of each boosting step α_k at each step, such that

$$\hat{f}(x) = \sum_{k=1}^{K} \alpha_k f(x)_k. \tag{1.5}$$

With AdaBoost, it was shown that the boosting ensemble of models dramatically improved the performance from just using a single, weak learner (Bauer and Kohavi 1999; Breiman 1998; Meir and Rätsch 2003; Ridgeway 1999). An illustration of how a boosting ensemble works is given in Figure 1.1.

The idea of boosting got further generalized from the classification case with AdaBoost, to a general model who focuses on difficult observations by large residuals instead of only misclassifications. The first *gradient boosting* algorithm was presented in Friedman 2001, where the idea is to iteratively fit the weak learner to the gradient of a loss function computed at the previous iteration, which enables loss functions defined for e.g. regression, or to maximize a *likelihood.*

Maximum likelihood estimation (MLE) is the most popular methods for estimating model parameters θ within statistics (Casella and Berger 2021), and maximizes a likelihood function $\mathbb{L}(\theta|\mathbf{x}, y)$ so that the observed data D is most probable. It is easy to make a likelihood into a loss function, as minimizing the negative of the likelihood is the same as maximising it, $L_\theta(y, f(\mathbf{x}) = -\mathbb{L}(\theta|\mathbf{x}, y)$. In Paper III, we will see that being able to use boosting for maximum likelihood estimation is very handy when solving problems outside the common regression or classification cases.

An implementation of gradient boosting, which further attracted the attention from the machine learning community towards boosting due to its high

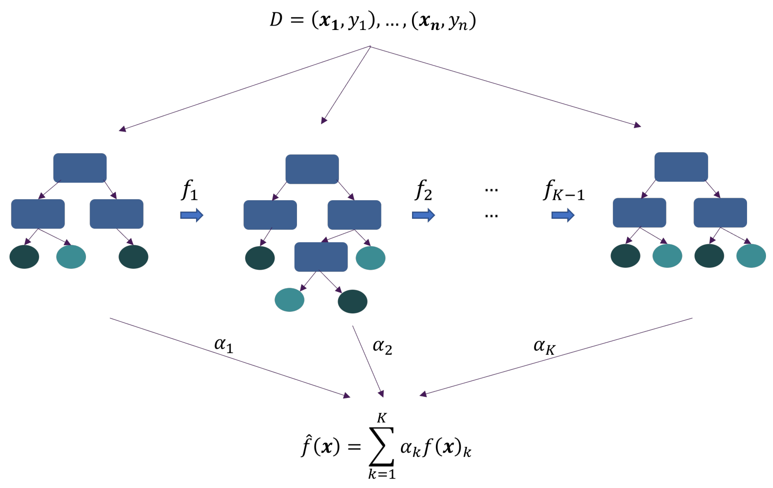

$D = (\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_n}, y_n)$
f_1
f_2
f_{K-1}
α_1
α_2
α_K
$\hat{f}(\boldsymbol{x}) = \sum_{k=1}^{K} \alpha_k f(\boldsymbol{x})_k$

Figure 1.1: Illustration of a boosting model with simple decision trees as base learner.

performance and fast computations, is *XGBoost* (Chen and Guestrin 2016). Since its release in 2014, XGBoost has been a very popular machine learning method, and it has an impressive winning record when it comes to machine learning competitions. A notable characteristic of XGBoost is that it uses, in addition to numerous computational tricks, a second order approximation of the loss function in its computations, which speeds up the procedure. XGBoost, with trees as the base learner, have several characteristics making it a very good model for complex, safety-critical applications, as it

- allows missing values, which can often be present for critical situations
- handles data of mixed types, which can often be present in complex tasks
- are quite robust to outliers
- are good at handling multicollinearity, which also makes it suitable as a time lagged model
- are in general more robust than neural networks, due to the variance being decreased with the ensemble method
- are in general more explainable than neural networks, since it is a generalized additive model (GAM), which is by definition interpretable. However, the interpretability lowers with the size and number of base learners

This introduction will not go further into details of gradient boosting, as it is explained in more detail in Paper II and Paper III, where XGBoost is used for regression, classification and for maximum likelihood estimation in conjunction with time-to-event analysis.

1.1.3 Neural Networks

A highly popular alternative to boosting algorithms is neural networks, also called *artificial neural networks*, or *deep learning*. The development of neural networks began already in the 1940s (Andina et al. 2007), where the first systematic studies were published in McCulloch and Pitts 1943 and Pitts and McCulloch 1947, who created a computational model for neurons. The works focused on the behaviour of a single artificial neuron, who mimicked the behaviour of biological neurons. Inside the neuron, the inputs x_i are multiplied with weights w_i and transformed by a nonlinear activation function σ. This lead to what became the functional neurons,

$$f(\mathbf{x}) = \sigma(\sum_{i=1}^{n} w_i x_i + b), \tag{1.6}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $f(\mathbf{x})$ is the output from the artificial neuron. The first works on neural networks used a hard threshold value for the activation function σ, meaning the neuron activates for sums over a certain threshold or bias b, and provides a positive value. If the neuron is not activated, it provides a negative or zero value. An illustration of a artificial neuron is given in Figure 1.2.

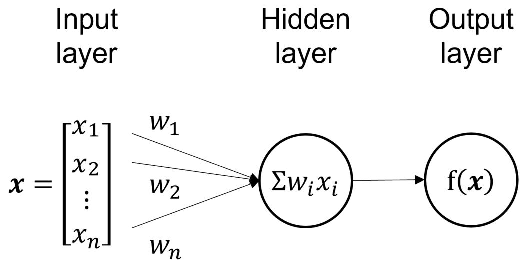


Figure 1.2: Illustration of a single artificial neuron

Even though a single neuron can perform only very simple tasks, the strength of neural networks lies in the connectivity of many neurons in a network. The first neural networks are those referred to as 2-layer where the the neurons are called the nodes. The input layer consists of the input data $\mathbf{x}$, which is connected with one layer called a *hidden layer*, with m number of nodes. An output layer is connected to the hidden layer, which becomes linear combination of the neurons of the hidden layer, and can model an output $\mathbf{y} \in \mathbb{R}^p$. A 2-layer neural network is illustrated in Figure 1.3.

Later it was shown that neural networks with simple activation functions, such as threshold functions, had several limitations, making them not useful

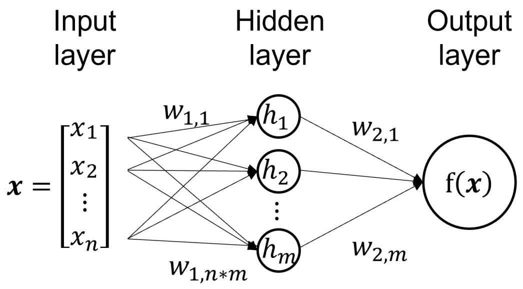


Figure 1.3: Illustration of a 2-layer neural network.

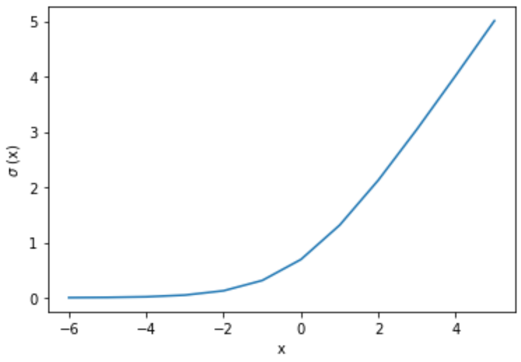


Figure 1.4: A Softplus function.

for real, complex data. This was solved by using "soft", derivable, nonlinear activation functions for σ instead of the hard threshold. Typical activation functions for neurons are Step, Linear, Sigmoid, TanH, ReLU or Softmax. In our neural network in Paper IV, the used activation function for the hidden layers is Softplus

$$a(z) = \frac{1}{\beta}\log(1 + e^{\beta z}) \tag{1.7}$$

which is a smooth approximation to the ReLU function, and is shown in Figure 1.4. However, in the simulations in Paper IV, the Sigmoid function is used.

With further development, the multi-layer neural network was created (Ivakhnenko and Lapa 1965; Ivakhnenko 1971; Schmidhuber 2015), which started the deep learning journey by the Multi-Layer Perceptron(MLP). A MLP is the same as a 2-layer neural network, however the input layer is connected in sequence to the l number of hidden layers, where the number of nodes m_l in each hidden layer may vary. The function for the output of the MLP becomes

$$\hat{f}(\mathbf{x}) = f_l(f_{l-1}(\cdots(f_1(\mathbf{x})))). \tag{1.8}$$

where f_j is a function of the outputs from layer $j \in [1, l]$.

Even though the foundational idea of artificial neurons and neural networks was developed, it lacked the theory on how to find the parameters in the neural networks, $\theta = \{\mathbf{w}_i, b_i\}_{i=1}^{m}$. Finding the optimal parameters is referred to as either training the network, or network learning, and several advancements where done on this topic. Hebb, D.O. (Hebb 2005) introduced *Hebbian Learning* in 1949, which were later simulated by Farley and Clark 1954. In Hebbian learning, the weight between two neurons increases if the neurons activate at the same time, and decreases if they activate separately, meaning the networks is trained in an unsupervised manner. Next, the network learning was taken into the supervised learning setting by Rosenblatt 1958, which also uses the network output to train a *perceptron*, i.e. a Feed Forward Neural Network. *The perception rule* takes in training data $(\mathbf{x_1}, y_1), \cdots, (\mathbf{x_n}, y_n)$ and updates the weights according to

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \alpha(y_k - \hat{f}(\mathbf{x}_k))\mathbf{x}_k \tag{1.9}$$

where k is the updating step, α is the *learning rate* controlling the magnitude of the updates, and $f(\mathbf{x}_k)$ is the current prediction from the output layer. This was further adapted to other loss functions, namely Least Mean Square and Sum of Square Errors in Widrow and Hoff 1960. However, efficient training for neural networks with multiple layers was still missing. Therefore, it was the use of *gradient descent* who made the neural network training effective (Bryson and Denham 1962; Kelley 1960; Pontryagin 1987), namely a first-order iterative optimization algorithm for finding the local minimum of a differentiable function. The idea behind gradient descent is to take small steps in the opposite direction of the gradient of the loss function at the current point, since this will be the direction of the steepest descent

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \alpha \nabla L(y_k, \hat{f}(\mathbf{x}_k)). \tag{1.10}$$

Finding the steepest descent in a chain of several layers can be done by iterating the chain rule of differentiation. This is called *backpropagation*, where the gradient of the loss function is computed with respect to each weight for one layer at the time, iterating backwards from the last layer, by the use of the chain rule

$$\frac{\partial L(y, \hat{f}(\mathbf{x}))}{\partial \mathbf{w}_j} = \frac{\partial L(y, \hat{f}(\mathbf{x}))}{\partial f_j(\mathbf{x})} \frac{\partial f_j(\mathbf{x})}{\partial \mathbf{w}_j} = \frac{\partial L(y, \hat{f}(\mathbf{x}))}{\partial f_l(\mathbf{x})} \frac{\partial f_l(\mathbf{x})}{\partial \mathbf{w}_{l-1}} \cdots \frac{\partial f_{j+1}(\mathbf{x})}{\partial f_j(\mathbf{x})} \frac{\partial f_j(\mathbf{x})}{\partial \mathbf{w}_j}, \tag{1.11}$$

where $\mathbf{w}_j$ are the parameters of layer $j = 1, \cdots, l$, with output $f_j(\mathbf{x})$. In this way, updating the parameters of a singly layer only needs the calculations of $\frac{\partial f_j(\mathbf{x})}{w_j}$ and $\frac{f_j(\mathbf{x})}{f_{j-1}(\mathbf{x})}$ (Kvamme 2020).

What neural networks lacks in stability and robustness, it makes up for in accuracy. Therefore, neural networks is a commonly used algorithms, also for safety-critical applications. Neural networks are especially strong for data types other than tabular data, such as images, video, text and time series, due to recent advancements such as convolutional layers, and architectures adopted to handle sequential data, which makes it very suitable for time series data.

1.2 Time series analysis

The analysis of time series data is a very important topic in many fields. Examples are economics and stock marked predictions, biomedicine and the effect of a treatment over time, reliability engineering an the degradation of engine parts over time, or environmental analysis and observations in relation to global warming. In the newer years, the analysis of sequential data such as video, text and speech analysis has become very important within computer-human communication and Natural Language Processing (NLP). Being able to create good models for data which evolves over time is therefore highly sought after.

However, the analysis of data points observed at different time points leads to a unique type of problems when working with statistical analysis and predictions. When sampling data from adjacent observations over time, it is clear that this creates data points with a certain dependency, whereas most statistical framework assumes that the data points are independent and identically distributed. In other words, a special set of statistical framework which allows dependent, adjacent observations have to be used when working with times series data.

1.2.1 Continuous stochastic process

When modelling the changes in a system over time in fields such as finance and physics, one often assume that the observed values are coming from dynamical systems such as *ordinary differential equations* (ODE) or *stochastic differential equations* (SDE). The former assumes a deterministic model for the changes based on the system's derivative

$$\frac{dy}{dt} = f(x), \tag{1.12}$$

while SDE has an additional stochastic term, resulting in a solution which is a *stochastic process*, which means that the system includes some randomness. An SDE can be written as

$$dx_t = f(x_t, t)dt + g(x_t, t)dW_t \tag{1.13}$$

or in its integral version

$$x_{t+k} - x_t = \int_t^{t+k} f(x_s, s)ds + \int_t^{t+k} g(x_s, s)dW_s. \tag{1.14}$$

Here, f is the drift term, which controls the expected change in x_t, while g is the diffusion term, i.e. the standard deviation of the change. SDE's are highly used to model dynamical systems with high uncertainty, such as stock prizes or physical systems such as heat flow or gas molecules.

Another type of continuous dynamical systems, while less used, is delay differential equations (DDE) as well as stochastic delay differential equations (SDDE), where the current state is expressed in terms of the derivative of a function of previous times, not only the current time

$$x_t = \eta(0) + \int_0^t f(s, \Pi(X_s))ds + \int_0^t g(s, \Pi(\eta(0)))dW(s), \quad t \geq 0 \tag{1.15}$$

where $\Pi : C \to \mathbb{R}^{dp}$ is a the initial condition consisting of previous states $\eta : \Omega \to C$ of the system

$$\Pi(\eta) \coloneqq (\eta(u_1), \ldots, \eta(u_p)) \in \mathbb{R}^{dp}, \tag{1.16}$$

where $u_1, \ldots, u_p \in [-\tau, 0]$ is time steps for the previous states. DDEs and SDDEs assumes, in other words, that the evolution of a system at a the present time depends on the past history, and they are reffered to as having *memory*. The use of previous states to find the present resembles the use of lagged values in autoregressive models, which will be further explained in the discrete time models. Models with memory have the potential to capture long-term trends instead of only the current changes. They are therefore used in many areas such as life sciences, population dynamics, physiology and also neural networks (Rihan 2021). More details about SDDE's are given in Paper IV, where they are combined with neural networks to model complex, dynamical systems.

1.2.2 Discrete data analysis

Many collections of time series data are assumed to come from a continuous stochastic process, however the series are most of the time approximated by a discrete time series due to restrictions in the collection or estimation methods. Discrete data sampled from a time series takes the form

$$D = \{\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \cdots, \mathbf{x}_{t_n}\}, \mathbf{x}_{t_k} \in \mathbb{R}^m \tag{1.17}$$

where $t_1 < t_2 < \cdots < t_n$, meaning the observations follow sequentially in time. The time step between two data points $\Delta t = t_{k+1} - t_k$ is often constant, however it is not always the case.

Due to the importance of time series data, there has been an exhaustive number of models and methods to perform time series analysis. However, all methods can in general be divided into two areas, methods in the *frequency domain* and in the *time domain*. Whereas the frequency domain involves wavelet and spectral analysis, the time domain involves using auto-correlation, meaning the correlation of an observation with delayed (previous) versions of the observation. In addition, the time domain can also involve cross-correlation, meaning the correlation of an series with delayed observations of other series/variables. The papers within this thesis sticks to the time domain methods.

When creating models for time series data, the goal is often to predict the state of the process some N time steps into the future, by using the current and past states. This is referred to as *forecasting*. One of the most common ways of working with time series modelling is by assuming that the current state of the system can be modelled as a linear combination of its previous states in addition to a stochastic term, which is called an *autoregressive model*. The autoregressive model AR(p) is defined as

$$x_t = \sum_{k=1}^{p} \phi_k x_{t-k} + \epsilon_t \tag{1.18}$$

where ϕ_k are the constants or parameters of the model, p is the number of time lagged observations used and $\epsilon_t \sim N(0, \sigma_\epsilon)$ is white noise. Modelling times series with autoregressive models can also be done for multivariate models, meaning models with more than one variable, and is then referred to as a vector autoregressive VAR(p) model

$$\mathbf{x}_t = \sum_{k=1}^{p} A_k \mathbf{x}_{t-k} + \epsilon_t \tag{1.19}$$

where $\mathbf{x}_t \in \mathbb{R}^m$ and $A_k \in \mathbb{R}^{m \times m}$. However, the autoregressive models are limited to the case of linear relationships. In many real world cases, making a linear approximation might not be appropriate. This problem is addressed in Paper III, by instead assuming an autoregressive boosting model instead of a linear, as well as in Paper IV, where neural networks are used.

1.2.3 Time-to-event analysis

Time-to-event analysis is a sub-field both within survival analysis and reliability theory. The analysis of time-to-event data is an important subject in application areas such as biomedicine, engineering, economics, ecology and agronomy. As given in its name, the goal is to predict the time until a certain event occurs. Thereby, time-to-event analysis can be seen as a regression problem, which evaluates the effect of covariates on the response, namely the time. However, time-to-event analysis is often complicated with the presence of censoring, meaning that true event time is not known for all observations. Instead, we only observe a higher and/or lower bound of the time. Paper III addresses the case of right censoring, meaning we only observe a lower bound for the censored observations,

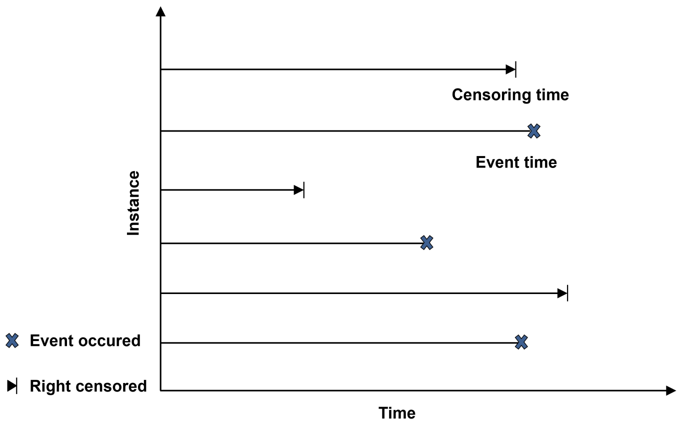


Figure 1.5: Illustration of time-to-event data with both event time observations and censored observations

as illustrated in Figure 1.5. Right censoring typically happens if a patient drops out of a study before achieving the test result, or a machinery component is replaces before its failure. An observation within time-to-event data consists of $(t, \delta, \mathbf{x}), \in \mathbb{R}^m$, where t is the event time or censoring time, dependent on which comes first, and δ is the censoring indicator such that $t = \min\{T, U\}$ and $\delta = \mathbb{I}(T \leq U)$.

The presence of censored observations makes statistical learning methods which requires training on a true, fully observed response variable unsuitable. Therefore, special types of models which handles a censored response has to be used, such as the Cox proportional hazard model or the Accelerated Failure Time model. The fitting of these regression models often involves the maximum likelihood estimation, which is applied in relation to the *survival function*. If the event time T has probability density function $f(t)$, the survival function is given as

$$S(t) = 1 - F(t) = P(T > t) \tag{1.20}$$

where $F(t) = \int_0^t f(u)du$.

As mentioned earlier, the likelihood function $\mathbb{L}(\theta|\mathbf{x}, y)$ is the conditional probability of the data D given the model parameters θ. Therefore, one can model the likelihood for a data point as the product of the probability that we will observe the true event time, $\Pr(T = t, U > t|\mathbf{x})$ and the probability that we observe the censoring time $\Pr(T > t, U = t|\mathbf{x})$, such that

$$\mathbb{L}(\theta|\mathbf{x}, y) = \Pr(T = t, U > t|\mathbf{x})^{\delta}\Pr(T > t, U = t|\mathbf{x})^{1-\delta}. \tag{1.21}$$

This likelihood can be used as a loss function for machine learning methods, by minimizing the negative of the likelihood, which will be explained in more detail in Paper III.

1.3 Motivating cases

The methods and models developed during this thesis are inspired by to specific use cases, where both are related to time-dependent environmental variables with high uncertainty, and where wrong predictions might provide major consequences. Details about the two use cases are provided in this Chapter.

1.3.1 Airplane landings on slippery runways

Weather conditions such rain, snow, wind and sleet can create difficult landing conditions for airplanes, especially during the winter season for airports located at areas with colder climates. When the runway is wet or covered with contamination such as snow and ice, the available friction between airplane tyres and the runway surface is significantly reduced. If the landing airplane is not able to retrieve enough braking force from the runway, it might not be able to stop properly, which can in the worst cases lead to runway overruns, which can again lead to damages, injuries and even loss of lives.

In order to apply appropriate braking action when landing, pilots need accurate and updated information about the runway conditions. This information is provided to pilots on a scale from 0-6 (earlier 0-5), ranging from *poor* to *good*, which corresponds to intervals of measurements or estimates of the *airplane braking coefficient* μ_B. The airplane braking coefficient is the ratio of vertical forces on the airplane wheels on the horizontal forces

$$\mu_B = \frac{F_R}{F_N} \tag{1.22}$$

where F_R is the frictional force, and F_N is the normal force working from the ground to the tires due to gravity, such that $F_N = mg\cos\theta - L$ where m is the airplane mass, g is the gravitation constant ($g \approx 9.81$),θ the angle of the ground and L is the aerodynamical lift acting from the air on airplane. These acting forces are illustrated in Figure 1.6.

The airplane braking coefficient is a dimensionless scalar which represents the properties between the two objects that are causing the friction, and will be low for slippery materials (approx. 0.16 for rubber and ice) and higher for materials with the potential to create a higher friction force (approx. 0.93 for rubber on dry concrete) (Wallman, Wretling, and Öberg 1997). Therefore, the airplane braking coefficient provides a good measure for the available potential for deceleration from the braking airplane wheels on the given surface conditions. When the braking coefficient is too low, airplane wheels will not manage to get enough friction force to brake and will begin to slip on the surface, potentially creating dangerous situations.

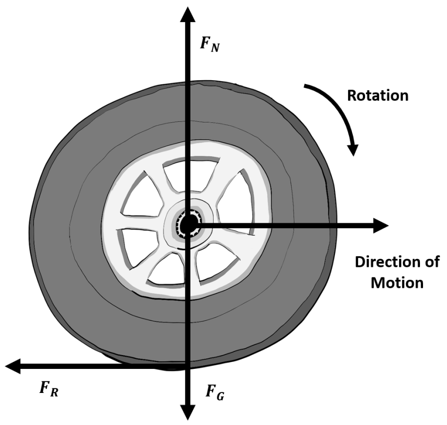


Figure 1.6: The forces acting on a landing airplane wheel. $F_N = mg\cos\theta - L$ is the normal force, $F_G = -mg\cos\theta$ is the gravitational force and F_R is the friction force due to braking

Unfortunately, measuring the runway friction in a way that corresponds to the experienced braking coefficient for landing airplanes, with a satisfactory precision, is difficult. While many different measurement devices have been developed, it is hard to find equipment that produces stable and consistent results. Moreover, in order to measure friction, the runway needs to be closed for traffic. Thus, such measurements cannot be carried out very frequently, if severe delays should be avoided. Therefore, high-accuracy, real-time prediction models for the braking coefficient is sought after.

The factors that mainly affects the runway surface conditions is naturally the weather conditions, and to provide correct decision support for the airport operators and pilots, one should create a model the effect of weather on the surface friction, considering both the present weather as well as interaction of weather variables over the previous time. However, the complexity and non-linearity of the physical relations controlling the surface friction, and their dependency on each other through time, makes it difficult to provide precise physical models for the braking coefficient. High accuracy predictions is espeically important in such a safety-critical situation, where a wrong decision might lead to major consequences and even loss of lives. Machine learning have on several occasions shown to be able to model complex physical phenomena with good performance, making it a good option for modelling the behaviour of physical phenomena such as snow and ice.

1.3.2 Stratospheric warmings

An exhaustive understanding of surface weather dynamics is crucial in a wide range of industry and societal sectors, such as planning marine operations, flights or farming, or managing energy assets (Eggen et al. 2022). However, the environment of the earth is a complex, nonlinear dynamical system with lot of uncertainty, limiting the time ahead one can make suitable weather predictions. Therefore, scientist are always on constant search better models and more relevant sources of information, which might lower the epistemic uncertainty of weather predictions.

Even though it is mainly of interest to predict the weather in the layer of the atmosphere we live in, namely the troposphere, the dynamics of troposphere is affected by the dynamics in the layers above. The second atmospheric layer, the stratosphere, lays around 15 km to 50 km above the surface. The dynamics in the stratosphere is connected to the dynamics of troposphere, and can affect the predictability of the weather on the surface (Baldwin and Dunkerton 2001; Butler et al. 2019). Hence, better understanding and modelling of stratospheric weather has the potential to enhance surface weather prediction.

One very clear example of stratospheric influence on surface weather is the extreme events called sudden stratospheric warmings (SSW), where a sudden disruption in the winter circulation of the stratosphere happens, followed by a increase in the temperature of several tens of degrees. This event can influence the troposphere, such that shifts in the jet stream and storm tracks might increase in probability. This event can create harsher weather in the northern parts of the globe, which might affect several sectors in society and industry (Eggen et al. 2022).

The beginning of a SSW is very visible in the stratospheric wind. At both of Earth's polar regions, there is a circulation of winds in the stratosphere, which is called a polar vortex, and is illustrated in Figure 1.7. The direction of the circulation in the polar vortex is called the U wind. As explained in A. Karpechko, Tummon, and Secretariat 2016 and Hitchcock and I. R. Simpson 2014, it is particularly interesting to study the U wind component. The U wind has a seasonal pattern, and has a positive direction in winter time (rotates counter clockwise), but changes direction in the summer time. However, when a SSW occurs, we can observe that the direction suddenly changes for a small amount of time in the winter season, and goes clockwise. Therefore, if we are able to make predictions about the U wind component in the stratosphere, we might be able to predict the potential occurrence of a SSW. However, making real-time measurements of the stratospheric weather is not an easy task due to physical limitations of measurement equipment at such a high altitude, making it sought after to have reliable forecasts of this weather phenomena.

1.4 Challenges with data driven methods for safety critical systems

As machine learning and data driven methods are changing the way we model and interact with the world, it begins to affect critical tasks such as engineering, health, environment and transportation. Thereby, taking safety and trust into account is getting very important. Especially within the area of safety-critical systems, where any system failure or wrong decision can lead to disastrous evens

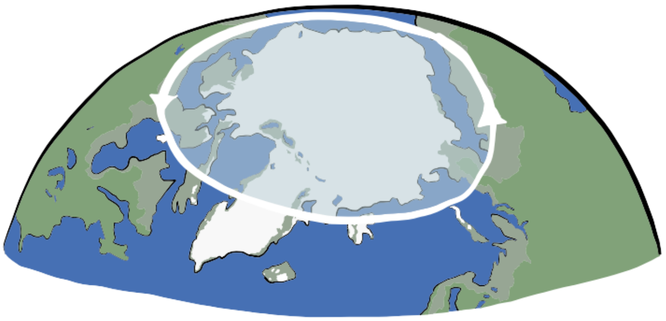


Figure 1.7: Illustration of the polar vortex at the north pole during winter time, with a positive U wind direction.

such as serious equipment damage, harm to the environment or even injury or loss of human lives (Tambon et al. 2022).

The term safety is used across a large number of fields, and refers to the absence of failures or conditions which make a system dangerous, such as having safe food and water, safe airplane trips, safe industrial plants, safe medical treatment and safe living environment (Varshney 2016). What makes each system safe varies from case to case, and has to be assessed individually for each field. However, in general, providing safety involves identification of potential hazards, assessment of their risk and implementation of strategies to minimize either the probability of the hazard or the consequences if it occurs. In other words, we want to minimize the risk and *epistemic uncertainty* associated with unwanted outcomes with severe and harmful consequences (Möller 2012). Epistemic uncertainty refers to uncertainty about a system which is present due to lack of knowledge, and the success of the safety analysis is inherently dependent on how good our knowledge is about the complete aspects of the system. Therefore, epistemic uncertainty is key part of safety, since harmful outcomes often occur in situations and operating conditions that are rare, unexpected, and might not have been observed earlier. More details about epistemic uncertainty is provided in this Chapter.

Many safety-critical systems are very complex, consisting of numerous components and parameters which can have complicated interactions and effects on the system state. This is where machine learning becomes a useful tool, as it have the ability to model very complicated and non-linear systems with many variables, as long as it gets enough data with good quality in the training phase. In this way, machine learning can provide good decision support by providing more knowledge about the systems, such that uncertainty is lowered.

In the later years, machine learning has also began interacting directly with the physical world. Examples of this is autonomous cars and ships, assisted living, medical devices and aviation software (Pereira and Thomas 2020). This unlocks even more challenges, as it excludes the governing and domain knowledge of human operators, and one have to completely rely on the decisions from the machine learning systems. This makes it even more important to have reliable and trustworthy AI systems which can handle the unexpected.

In the rest of this chapter, we go through some challenges with data driven methods for safety critical systems, and how the papers in this thesis address them.

1.4.1 Uncertainty

When working with safety-critical systems, its important to be able to trust predictions and decisions, which means one should be provided information about when to not trust a system. If an operator of a critical system is given information about the *confidence* in the predictions, the operator can decide to be critical about or ignore the prediction, and might seek other sources of decision support. If an automated AI system is uncertain about its prediction, it can forward the decision to a human operator.

Machine learning models, as well as other models and systems, do not always provide the correct prediction. There can be several reasons for this, such as noisy data or unknown/unexpected input data that is outside the trained hypothesis space. Many prediction systems, such as those created based on physics or human knowledge, often provide some information about the confidence in the system, including information about their own limitations. A human domain expert can say that "this is outside my expertise", and a physics-based model can often tell within which boundary it is valid. However, most current machine learning models, on the other hand, fail to properly assess when “they do not know” (Tambon et al. 2022). To make informed decisions, machine learning models should not only aim at providing high-performing predictions, but also describe as precisely as possible the uncertainty remaining in their outcomes (Kläs and Vollmer 2018).

Uncertainty estimates addresses the ability of a model to acknowledge its own limitations, and provide information about the confidence in its prediction. Within machine learning, uncertainty is often categorized as either aleatoric or epistemic uncertainty, and Paper IV focuses on predicting both types of uncertainty.

1.4.1.1 Aleatoric uncertainty

Aleatoric uncertainty quantifies the stochasticity that is inherent in data, and can be generated e.g. due to noisy sensors, which will be discussed further in in Section 1.4.2, or natural randomness such as a coin flip or movement of gas particles. The aleatoric uncertainty is therefore regarded impossible to reduce by providing any additional information, such as more data or variables, as the uncertainty will always be present in the data.

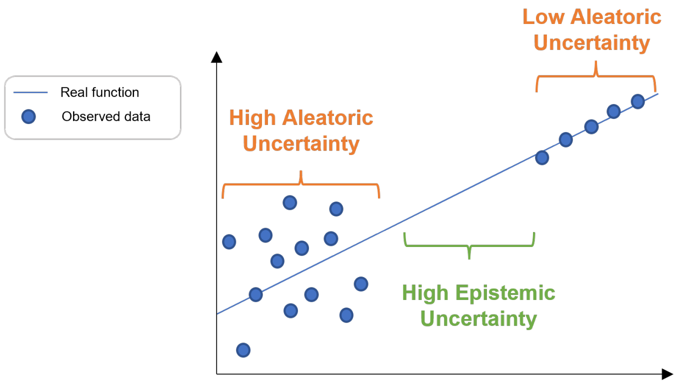

Real function
Observed data
High Aleatoric Uncertainty
Low Aleatoric Uncertainty
High Epistemic Uncertainty

Figure 1.8

1.4.1.2 Epistemic uncertainty

As mentioned earlier, epistemic uncertainty quantifies the uncertainty due to lack of knowledge about the system in scope. Within machine learning, a large reason for epistemic uncertainty is the finite amount of data used in training the model. In other words, the models knowledge about the system is restricted to the situations it has observed earlier, and since the amount of data used in training is limited, it has rarely seen all possible states a system can be in. Especially, we often lack observation from the safety-critical situations, which are the ones important to model correctly. Figure 1.8 illustrates the difference between aleatoric and epistemic uncertainty.

The epistemic uncertainty is regarded as being reducible, by gaining more knowledge. This could either be extending the number of data points, getting more quality or relevant data, add new information such as new variables or by providing more training data from the rare, often more critical, states. However, gaining more data is rarely a simple task, especially from high risk situations. Therefore, the quantification of epistemic uncertainty is important, and the epistemic uncertainty can again be divided into two categories, model uncertainty and approximation uncertainty.

Model uncertainty may occur when either the model function or the training algorithm have limitations, preventing the trained model from properly modelling the true underlying function. In other words, the hypothesis space available for the model to explore does not cover the true hypothesis space of the function. An example of this is given in Figure 1.9, where a linear model is fit to non-linear data, showing that a model assuming linearity is not complex enough to capture the pattern in the data created by the real function.

Approximation uncertainty may occur when the model is not trained well enough on all possible situations, i.e. not trained well enough within the hypothesis space. This happens due to lack of training data, e.g. if we do not have a large enough amount of data or does not have high enough data quality. In addition, if the given training data does not contain observations representative

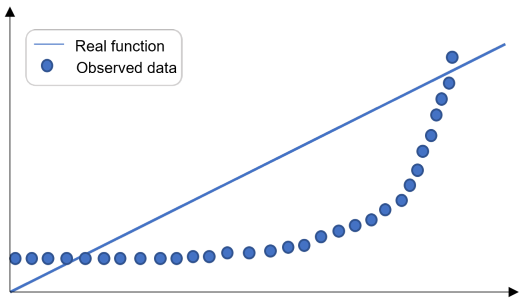


Figure 1.9

for all possible situations, the trained hypothesis space will not cover the whole function space, such as shown in the regions in the middle of figure 1.8. This makes approximation uncertainty relate closely to out-of-distribution (OOD) detection. OOD data refers to data points which highly differ from the data used to train the machine learning model, where unreliable and unsafe prediction is prone to happen. OOD data can often occur when a model is put into production, such as wrong measurements or unexpected events, where the latter might be extra critical to handle correctly. OOD data can also occur if the model is deployed in another environment or context than what it was trained for, such as a new calibration of the sensors. Therefore, OOD detection is an important task within safety-critical systems, to give warning to not trust the system for these situations, as the epistemic uncertainty is high. More discussions on OOD data and data challenges are provided in the following sections.

1.4.2 Data management

The development of a machine learning model differentiates from the previous methods for model development, which were more dominated by well-defined rules such as laws of physics or explicitly defined algorithmic. Machine learning algorithms learn on their own, which means their models are created almost entirely of data. As a consequence of this, any challenges related to the data source and data management also becomes challenges for the machine learning model. When working with safety-critical systems, there exists several difficulties related to gathering and managing data, and we will mention some of the challenges which are addressed in this thesis.

1.4.2.1 Data quality

Data collected are limited in their accuracy, and can be affected by different types of quality issues. During data acquisition, if a problem on data sources occurs which compromises its quality, it can produce noisy and unreliable data, which might limit the useful information that is sensed. When it comes to e.g. sensor data, it can be lost, corrupted or even acquired from different sources if sensors are replaced or calibrated some times during the gathering phase. The *sensor accuracy* might have a large impact of the behaviour of the machine learning models, and some influencing sensor characteristics are sensing mode, range, resolution, noise characteristics, calibration, and placement (Kläs and Vollmer 2018). Sensor accuracy is one challenge explored in Paper I, where different kind of sensor measurements from landing airplanes are previously shown to be unreliable, and the paper discusses methods to make these noisy measurements useful for estimating the airplane braking coefficient. If it is not possible to remove the source of noise in the measurements, it will be regarded as adding aleatory noise to the data.

1.4.2.2 Annotation

Within machine learning, having reliable data for the response variable is especially important, as the algorithms treat this as the ground truth to be optimized on during the training phase. During *annotation*, meaning labelling the response correctly in conjunction with the wanted prediction, quality can be affected. In Paper II,t the wanted response variable, the airplane braking coefficient, is subject to measurement noise, as explored in Paper I. Additionally, the response variable is subject to errors in an interpolation algorithm turning sensor measurements into estimates of the braking coefficient. This means that that the added aleatory noise will create an upper bound for the accuracy possible to achieve from any models trained on this data, and if the noise is to large, the algorithm might fail to discover anything useful at all (Pereira and Thomas 2020).

1.4.2.3 Censoring

One additional problem for the response variable, is that the correct label/value is not always possible to retrieve. In paper I and II, we show that true value for the response variable μ_b (the braking coefficient) is known only for for approx. 3-4% of the data points. Instead, we observe a measurement of the *used* braking coefficient μ_{used} by the landing airplane for the rest of the data, which is actually a lower bound of the true available braking coefficient, $\mu_{\text{used}} < \mu_B$. Only observing a lower bound of the true response is a commonly known problem refereed to as right censoring as explained in Section 1.2.3. Right censoring can occur within risk critical problems such as predicting time until failure of a system, where there might exist only a few, if any, data from the situation where the system actually failed. Optimizing on a lower bound instead of the exact value puts a lot of restrictions on the loss function and methods of optimization. Nevertheless, removing the censored data points is often not an alternative. Due to lack of data for the critical situations, the lower bound might be almost all information available of these situations. Paper II addresses the problem of censored data by using two different predicting models; one model to predict censoring of data and one model trained solely on uncensored data. Paper III addresses the issue of right censoring more generally, by introducing a novel boosting model.

1.4.2.4 Inadequate data distribution

Another reason why removing the censored data points is rarely a good alternative, is that this intervention might result in a biased machine learning model. When certain data points are absent in the training dataset, the model will not learn about the whole situation in the correct way, since the data set is not representative for the entire system it was gathered from. In Paper II we create two models trained on different parts of the data, as mentioned in the previous section. This is done as using one model trained solely on uncensored landings will provide a system which does not have proper understanding of all possible states of the runway conditions. Instead, it will only understand the situations covered by the 3-4% landings with lower friction coefficients. In other words, the training distribution would not be the same as the real system distribution. Since machine learning algorithms often assume that training and operating data are drawn from the same distribution, the algorithms can be very sensitive even to small changes in distribution.

1.4.2.5 Lack of data for critical situations

The lack of data from critical situations is one of the most common problem when working with data driven methods for safety critical systems. As it might be very damaging or dangerous to be in the critical states, one might never have observed these situations, and therefore lack the data. Typical machine learning techniques, only learning from the data, might not be able to give any trustworthy predictions about the critical states, which is the ones highly important to identify. In other words, lack of data from the critical situations makes the machine learning models have a high epistemic uncertainty for these situations.

1.4.2.6 Inadequate performance evaluation metric

The definition of an appropriate performance evaluation metric is also important when working with machine learning, to be able to assess the performance of the model correctly. Performance evaluation metrics are used to estimate the performance of a model, which provides information about how well different models works the specific application cases. One classical examples of this is Root Mean Square Error (RMSE), which measures the Euclidean distance between the true and predicted value. If models are evaluated using the wrong metric, one might get the wrong impression on how well the model works for different possible scenarios. This can be compared to as a company selecting a key performance index to evaluate the state of the business, where one will get a complete wrong assessment of the company's economy if it is evaluated at wrong indexes. Not all types of data and applications have an intuitive correct way of measuring performance. The problem of performance evaluation metric is discussed in Paper II in relation to the very unbalanced data, as well as in Paper III, discussing the lack of evaluation metrics for dependent, right censored data points.

1.4.3 Interpretability

Machine learning models can become very complex, consisting of hundreds or even millions of parameters, making it incomprehensible for humans to understand how they make their predictions. However, there are a lot of reasons why it is important to have some understanding of how a prediction model works, especially when safety-critical systems are involved, such as:

- Verification. Trained machine learning models might not always behave as we expect them to. A classical example is the husky vs wolf example, where a neural network classifies images of husky dogs and wolves. However, the model bases its prediction not on how the animal looked like, but solely on if there was snow present in the image (Ribeiro, Singh, and Guestrin 2016). To verify that the machine learning models behave as we intend them to do, interpretability is important.

- Improvement. In order to improve our systems, it is important to understand the systems' weaknesses and flaws. In addition, it is important to find potential unwanted biases due to the collected data, in order for them to be removed.

- Learning. Machine learning might observe and find patterns and relationships not accessible by humans, which can provide us with more knowledge. To be able to learn from a system, we have to understand how it works.

- Justification. Prediction models need to give explanations of the reasons for their decision, especially if it is unexpected.In some areas, one might also need to provide justification in order to be compliant with legislation.

- Monitoring. If the reasons for a system's decisions are visible, it is also possible to monitor for potential error both in the model, but also in the data source the prediction is based on. Especially if the input data is outside the trained hypothesis space, the model might provide completely wrong predictions, which might be visible by looking at the reasons for the decision.

The increased use of black-box algorithms, and the challenges that accompanies it, has escalated the focus on creating Explainable Artificial Intelligence (XAI) and Interpretable Models. Within safety-critical applications, it is difficult to trust any decisions without knowing the arguments why the decision was made, meaning being able to verify and monitor that the system works properly for each decision. In Paper II, we create additional decision support by providing arguments from XAI methods for all predictions, to create a trustworthy decision support system. We also illustrate the increased value of predictions from machine learning models when XAI is added.

1.4.4 Model limitations from deployment

Another challenge that has to be addressed when putting a machine learning model in operation, is limitations in the deployed device such as computation power or efficiency. Many real-world task require fast computations in order to act on time, such as autonomous vehicles or stock market predictions. This adds limitations to the time a machine learning model can use for when providing predictions and uncertainty estimates. However, some of the most popular methods for estimating uncertainty in machine learning are based on Bayesian methods (e.g. Maddox et al. 2019; Teye, Azizpour, and Smith 2018), which apply a probability distribution over the parameters, instead of a single set of parameters. Then, the uncertainty can be retrieved by running a series of simulations on the model either with different priors, perpetuated input data, or by the approximation called Dropout (Gal and Ghahramani 2016), and then calculate the variance of the different runs. The limitation of the Bayesian approach is the requirement of many forward-passes to estimate the uncertainty, meaning it can be time consuming. It is therefore necessary to develop methods for uncertainty quantification which can happen fast, and this problem is addressed in Paper IV, in conjunction with both aleatoric and epistemic uncertainty.

1.4.5 Instability

Some machine learning models, such as neural network, tend to be very unstable, meaning small perturbations in the input data can create highly different predictions (Antun et al. 2020; Papernot and McDaniel 2018). One way to reduce the instability is to create ensemble of models, such as gradient boosting, which lowers variance and increases robustness. However, instability is one of the areas where purely mathematical or physics models some times have an advantage over learning models. Explicitly specified models, also referred to as forward modelling approaches, are specified from system's characteristics and knowledge and not purely on data. Therefore, small perturbations in data will not effect these models in the same manner.

Another positive characteristic of forward modelling approaches is that they are often more capable of informing about the boundaries they can operate within, which means that they can inform about their epistemic prediction uncertainty. In addition, since these models are not entirely specified by data, they can often provide more reasonable predictions outside the normal operating domains, meaning they might behave more stable and nicely for unexpected events and OOD data.

However, forward modelling approaches are limited by model simplifications, omission of information and difficulty in modelling complex systems with a satisfactory accuracy (Gardoni, Der Kiureghian, and Mosalam 2002). The desire to have the best properties from both data-driven methods and forward modelling approaches has in the recent years given attention to *physics-informed machine learning* (PIML). This means that the machine learning models are not trained unrestrictedly on data, but have some conditions put on the training either inspired by the laws of physics, classical mathematical models or empirical knowledge (Karniadakis et al. 2021). This can be very helpful in making the prediction models more robust to perturbations in the data, provide better estimates for uncertainty and use restrictions on their behaviour on OOD data. In addition, PIML models can potentially be better trained on fewer data points, since the constrictions limits the hypothesis space that can be explored. The machine learning model developed in Paper IV, the Delay-SDE-net, is a type of physics-informed neural network, as it is constrained to behave as a stochastic differential delay equation, providing it with many of the nice properties of explicit formulated models.

1.5 Summary of paper contributions

Paper I discusses some of the challenges with converting time series of measurements of flight data into reliable estimates of runway friction. We show that calibrating the acceleration in sensor measurements from landing airplanes has a clear effect on the estimated friction coefficient, which will in turn lead to an increased number of landings being classified as friction limited, meaning within the critical situations. This will effect the evaluated accuracy of runway reports and models, meaning the wrong calibration might lead to false conclusions on prediction performance and trustworthiness of methods to identify slippery landings.

Paper II presents a decision support system for airplane landings, which combines the predictions of runway surface conditions from time-lagged boosting models with explainable AI. We show that the boosting models perform higher than the previously used methods on Norwegians Airports, also compared to human assessment. This shows the strong abilities of machine learning to find and use patterns to model complex, physical phenomena when domain knowledge is included through the extraction of explanatory variables. We show that complex machine learning models are also capable of being more transparent than simpler methodology as scenario models and engineering-based models, as game-theory inspired explanability methods provides additional decision support in terms of providing arguments for and against its predictions.

Paper III illustrates a typical problem with data from safety-critical situations, namely the lack of data from critical situations. This can sometimes take the form as right censored data points. We address this problem by developing a novel model for dependent censored data, which often appears in risk analysis and time-to-event data. The developmed model, Clayton-boost, shows a strong ability to remove prediction bias in the presence of dependent censoring, and outperforms the typical used methodology when the dependency increases. Clayton-boost also performs very well on higher percentage censoring, where the commonly used methods tends to fail and highly overestimate the event times.

Paper IV introduces Delay-SDE-net, a novel neural network algorithm for time series prediction with uncertainty estimates, which is based on stochastic delay differential equations. The model fills a gap in the literature, as it is suitable for time series due to its autoregressive nature, can model complex structure since it consists of neural networks, and predicts both aleatory and epistemic uncertainty instantly. We derive the theoretical error of the Delay-SDE-net and analyse the convergence rate numerically. Additionally, the model is evaluated at both simulated data and a real-world case study, where measurements of stratospheric U wind and temperature are used to predict the future U wind. This case is highly relevant as the complexity of the Earth's environment creates a lot of uncertainty, and the prediction of stratospheric U wind can help identify sudden stratospheric warming and extreme weather at the surface. At comparisons with similar models, the Delay-SDE-net has consistently the best performance, both in predicting time series values and uncertainties

References

Andina, D. et al. (2007). "Neural networks historical review". In: *Computational Intelligence*. Springer, pp. 39–65.

Antun, V. et al. (2020). "On instabilities of deep learning in image reconstruction and the potential costs of AI". In: *Proceedings of the National Academy of Sciences* vol. 117, no. 48, pp. 30088–30095.

Baldwin, M. P. and Dunkerton, T. J. (2001). "Stratospheric harbingers of anomalous weather regimes". In: *Science* vol. 294, no. 5542, pp. 581–584.

Bauer, E. and Kohavi, R. (1999). "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants". In: *Machine learning* vol. 36, no. 1, pp. 105–139.

Breiman, L. (1996). "Bagging predictors". In: *Machine learning* vol. 24, no. 2, pp. 123–140.

— (1998). "Arcing classifier (with discussion and a rejoinder by the author)". In: *The annals of statistics* vol. 26, no. 3, pp. 801–849.

Bryson, A. E. and Denham, W. F. (1962). "A steepest-ascent method for solving optimum programming problems". In.

Butler, A. et al. (2019). "Sub-seasonal predictability and the stratosphere". In: *Sub-seasonal to seasonal prediction*, pp. 223–241.

Bzdok, D., Altman, N., and Krzywinski, M. (2018). "Statistics versus machine learning". In: *Nat Methods* vol. 15, no. 4, p. 233.

Casella, G. and Berger, R. L. (2021). *Statistical inference*. Cengage Learning.

Chen, T. and Guestrin, C. (2016). “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: Association for Computing Machinery, pp. 785–794.

Eggen, M. D. et al. (2022). “Stochastic modeling of stratospheric temperature”. In: *Mathematical Geosciences*, pp. 1–28.

Faria, J. M. (2018). “Machine learning safety: An overview”. In: *Proceedings of the 26th Safety-Critical Systems Symposium, York, UK*, pp. 6–8.

Farley, B. and Clark, W. (1954). “Simulation of self-organizing systems by digital computer”. In: *Transactions of the IRE Professional Group on Information Theory* vol. 4, no. 4, pp. 76–84.

Freund, Y. (1995). “Boosting a weak learning algorithm by majority”. In: *Information and computation* vol. 121, no. 2, pp. 256–285.

Freund, Y., Schapire, R. E., et al. (1996). “Experiments with a new boosting algorithm”. In: *icml*. Vol. 96. Citeseer, pp. 148–156.

Friedman, J. H. (2001). “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics*, pp. 1189–1232.

Gal, Y. and Ghahramani, Z. (2016). “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. PMLR, pp. 1050–1059.

Gardoni, P., Der Kiureghian, A., and Mosalam, K. M. (2002). “Probabilistic capacity models and fragility estimates for reinforced concrete columns based on experimental observations”. In: *Journal of Engineering Mechanics* vol. 128, no. 10, pp. 1024–1038.

Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*. Psychology Press.

Hitchcock, P. and Simpson, I. R. (2014). “The downward influence of stratospheric sudden warmings”. In: *Journal of the Atmospheric Sciences* vol. 71, no. 10, pp. 3856–3876.

Ivakhnenko, A. G. and Lapa, V. G. (1965). *Cybernetic Predicting Devices*. CCM Information Corporation.

Ivakhnenko, A. G. (1971). “Polynomial theory of complex systems”. In: *IEEE transactions on Systems, Man, and Cybernetics*, no. 4, pp. 364–378.

Karniadakis, G. E. et al. (2021). “Physics-informed machine learning”. In: *Nature Reviews Physics* vol. 3, no. 6, pp. 422–440.

Karpechko, A., Tummon, F., and Secretariat, W. (2016). “Climate predictability in the stratosphere”. In: *Bulletin nº* vol. 65, p. 1.

Kearns, M. and Valiant, L. (1994). “Cryptographic limitations on learning boolean formulae and finite automata”. In: *Journal of the ACM (JACM)* vol. 41, no. 1, pp. 67–95.

Kelley, H. J. (1960). “Gradient theory of optimal flight paths”. In: *Ars Journal* vol. 30, no. 10, pp. 947–954.

Kläs, M. and Vollmer, A. M. (2018). “Uncertainty in machine learning applications: A practice-driven classification of uncertainty”. In: *International conference on computer safety, reliability, and security*. Springer, pp. 431–438.

Kvamme, H. (2020). “Time-to-Event Prediction with Neural Networks”. PhD thesis. University of Oslo.

Maddox, W. J. et al. (2019). “A simple baseline for bayesian uncertainty in deep learning”. In: *Advances in Neural Information Processing Systems* vol. 32.

Mayr, A. et al. (2014). “The evolution of boosting algorithms”. In: *Methods of information in medicine* vol. 53, no. 06, pp. 419–427.

McCulloch, W. S. and Pitts, W. (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* vol. 5, no. 4, pp. 115–133.

Meir, R. and Rätsch, G. (2003). “An introduction to boosting and leveraging”. In: *Advanced lectures on machine learning*. Springer, pp. 118–183.

Möller, N. (2012). “The concepts of risk and safety”. In: *Handbook of risk theory: epistemology, decision theory, ethics, and social implications of risk* vol. 1, pp. 55–85.

Papernot, N. and McDaniel, P. (2018). “Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning”. In: *arXiv preprint arXiv:1803.04765*.

Pereira, A. and Thomas, C. (2020). “Challenges of machine learning applied to safety-critical cyber-physical systems”. In: *Machine Learning and Knowledge Extraction* vol. 2, no. 4, pp. 579–602.

Pitts, W. and McCulloch, W. S. (1947). “How we know universals the perception of auditory and visual forms”. In: *The Bulletin of mathematical biophysics* vol. 9, no. 3, pp. 127–147.

Pontryagin, L. S. (1987). *Mathematical theory of optimal processes*. CRC press.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). “" Why should i trust you?" Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.

Ridgeway, G. (1999). “The state of boosting”. In: *Computing science and statistics*, pp. 172–181.

Rihan, F. A. (2021). “Stochastic Delay Differential Equations”. In: *Delay Differential Equations and Applications to Biology*. Singapore: Springer Singapore, pp. 123–141.

Rosenblatt, F. (1958). “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* vol. 65, no. 6, p. 386.

Schapire, R. E. (1990). “The strength of weak learnability”. In: *Machine learning* vol. 5, no. 2, pp. 197–227.

Schmidhuber, J. (2015). “Deep learning in neural networks: An overview”. In: *Neural networks* vol. 61, pp. 85–117.

Tambon, F. et al. (2022). “How to certify machine learning based safety-critical systems? A systematic literature review”. In: *Automated Software Engineering* vol. 29, no. 2, pp. 1–74.

Teye, M., Azizpour, H., and Smith, K. (2018). “Bayesian uncertainty estimation for batch normalized deep networks”. In: *International Conference on Machine Learning*. PMLR, pp. 4907–4916.

Varshney, K. R. (2016). “Engineering safety in machine learning”. In: *2016 Information Theory and Applications Workshop (ITA)*. IEEE, pp. 1–5.

Wallman, C.-G., Wretling, P., and Öberg, G. (1997). *Effects of winter road maintenance: state-of-the-art*. Statens väg-och transportforskningsinstitut.

Widrow, B. and Hoff, M. E. (1960). *Adaptive switching circuits*. Tech. rep. Stanford Univ Ca Stanford Electronics Labs.

Papers

Paper I

Estimating Runway Friction Using Flight Data

Alise Danielle Midtfjord, Arne Bang Huseby

Published in *Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference*, 2020, DOI: 10.3850/978-981-14-8593-0.

Abstract

During the winter season, contamination of runway surfaces with snow, ice, or slush causes potential economic and safety threats for the aviation industry. The presence of these materials reduces the available tire-pavement friction needed for retardation and directional control. Therefore, pilots operating on contaminated runways need accurate and timely information on the actual runway surface conditions. Avinor, the company that operates most civil airports in Norway, have developed an integrated runway information system, called IRIS, currently used on 16 Norwegian airports. The system uses a scenario approach to identify slippery conditions. In order to validate the scenario model, it is necessary to estimate runway friction. The present paper outlines how this can be done using flight data from the Quick Access Recorder (QAR) of Boeing 737-600/700/800 NG airplanes. Data such as longitudinal acceleration, airspeed, ground speed, flap settings, engine speed, brake pressures are sampled at least each second during landings. The paper discusses some of the challenges with this. In particular, issues related to calibration of data are considered, and two different regression methods are compared.

Contents

The authors were partially supported by CERN.

I.1 Introduction

Slippery runways represent a significant risk to aircrafts especially during the winter season. Accidents, such as the Southwest Airlines jet skidding off a runway at Chicago Midway Airport in December 2005, as well as the similar accident with the Delta Connection flight at the Cleveland Hopkins International Airport in Ohio in February 2007, show that this is indeed a serious problem. For more details about the Chicago Midway Airport accident see Rosenker et al. 2007.

In order to apply the appropriate braking action, the pilots need reliable information about the runway conditions. Unfortunately, the accuracy of runway reports can sometimes be unsatisfactory. Having reliable methods for identifying slippery runway conditions is very important. However, measuring the runway friction with a satisfactory precision is difficult. While many different measurement devices have been developed, it is hard to find equipment that produces stable and consistent results. Another problem is that in order to measure friction, the runway needs to be closed for traffic. Thus, such measurements cannot be carried out too frequently. As a result, the runway reports are not as useful as one could hope. In particular, heavy snowfalls, or sudden drops in temperature may result in rapidly changing conditions. See Giesman 2005 and Rosenker et al. 2007.

Avinor, the company that operates most civil airports in Norway, initiated the so-called SWOP-project, a research and development project with contributions from the three airlines SAS, Norwegian and Widerøe. The main goal was developing methodology for predicting runway conditions utilizing weather data in addition to runway reports. Throughout two winter seasons various kinds of weather data were collected, such as air and surface temperature, humidity, precipitation, visibility and wind. Using these data, a scenario based weather model for slippery conditions was developed. A complete report from this project is given in Aarrestad et al. 2007.

Based on the SWOP-project, Avinor developed an integrated runway information system, called IRIS, currently used on 16 Norwegian airports. See Søderholm et al. 2009. IRIS consists of three parts: a *weather* model, a *runway* model and a *development* model. The weather model uses a scenario approach to identify slippery conditions. A description of an early version of this model can be found in Huseby and Rabbe 2008, while revised versions are presented in Huseby and Rabbe 2012 and Huseby and Rabbe 2018. The runway model uses mainly runway report data and assesses runway conditions on a five-level scale ranging from *poor* to *good*. See Huseby, Klein-Paste, and Bugge 2010 and Klein-Paste et al. 2012. The development model combines runway report data, precipitation and temperature data in order to issue warnings when the runway conditions are deteriorating.

The IRIS models have been verified by comparing the results to runway friction estimates based on flight data. The estimates are calculated by applying an airplane brake performance model developed by Boeing. A detailed description of this model is beyond scope of this paper, but some further details can be found in Klein-Paste et al. 2012. Instead this paper focuses on how to prepare

Table I.1: Number of landings and friction limited landings at Gardermoen airport and Tromsø airport.

Property	Gardermoen	Tromsø
Number of landings	228 700	19 278
Friction limited landings	8 342	5 762
% friction limited landings	3.6%	29.9%

the input to the model, and on the results that can be obtained.

I.2 Calculating the friction coefficient

The flight data, collected over ten winter seasons, from season 2009/2010 until season 2018/2019 is provided by Scandinavian Airlines Service (SAS) and Norwegian Air Shuttle AS and is gathered from the Quick Access Recorder (QAR) of Boeing 737-600/700/800 NG airplanes. We have available flight landings from 16 Norwegian airports, but in this paper, we only use flight landings at Tromsø and Gardermoen (Norway's largest airport). These airports are chosen since they have gathered data for the longest time. Besides Tromsø and Gardermoen are very different with respect to weather conditions, number of flights and runway operations. The final data set, containing only landings at Tromsø and Gardermoen, consists of approximately 248 000 flight landings, where the distribution is shown in Table II.2.

The airplane braking coefficient, μ_B, is calculated using the performance model developed by Boeing. This model uses the aircrafts' airspeed, longitudinal acceleration, gross weight, engine force, flap positions and deployment of thrust to calculate μ_B. The braking coefficient is used to represent the contribution of the wheel brakes to stopping the airplane, and is defined as the ratio of the stopping force contribution of the wheel brakes to the average airplane weight on wheels.

An important problem when analyzing flight data is deciding whether a landing is *friction limited* or not. Unless the pilot challenges the runway friction during the landing, the maximum friction available will not be utilized. In this case, μ_B reflect the amount of tire-pavement friction that was used. When wheel brakes are applied fully or to a high degree on slippery runways, the maximum attainable friction from the runway is used during the stop. In this case, the airplanes deceleration is limited by the friction available from the runway, and the obtained μ_B will reflect the amount of tire-pavement friction that is available. Such a landing is referred to as *friction limited*, and since the braking coefficient reflects the available tire-pavement friction, we refer to it as the *friction coefficient*.

To figure out if the brakes are applied fully during a landing, we check whether the brake pressure "requested" by the pilot exceeds the brake pressure corresponding to the measured deceleration. Whenever this occurs, the anti-skid system is activated, and all the available friction is used, i.e., the landing is friction limited.

As it is not possible to obtain a precise estimate of μ_B when a landing is not friction limited, only friction limited landings can be used for this purpose, i.e., 3.6% and 29.9% of the landings at respectively Gardermoen and Tromsø (Table II.2).

I.3 Calibration of acceleration measurements

In order to estimate the friction coefficient from the flight data, the acceleration of the landing aircraft is an important factor. Unfortunately, the measurement of the accelerations $a(t)$ is known to be biased. Thus, the true acceleration $\alpha(t)$ can be expressed with the following model:

$$\alpha(t) = a(t) + \epsilon. \tag{I.1}$$

The bias term ϵ is typically time-dependent, but since we consider a relative short time interval during a landing, the bias is assumed to be constant. To estimate the bias for each landing, velocities and positions are needed. However, GPS coordinates are sampled at different points of time than the other aircraft data. This may lead to time series of positions that are non-physical, as seen in Figure I.1. This is sometimes the case for the measured ground speed as well. These effects must be taken into account when velocities and positions are used together with other data. Accelerometer data is sampled with a higher sampling rate and is therefore a more reliable source than the GPS coordinates. As numerical integration is a well-behaved process in the sense integrals smoothen the result, this is a feasible method to obtain velocities and positions from the accelerometer data. In the next subsection we will show how the bias, ϵ, as well as the initial speed, v_0, can be estimated by combining accelerometer data and positions.

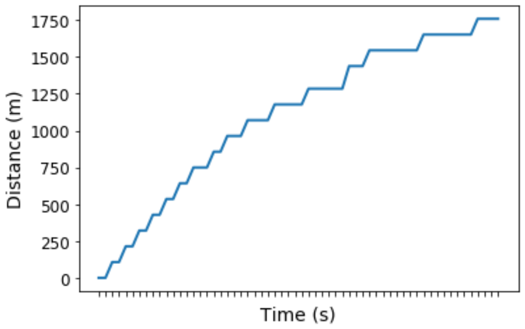


Figure I.1: A typical example of a time series of distances from the point where the aircraft touches the ground. The distances are calculated from the GPS positions recorded, and the asynchronous sampling between the GPS and the QAR makes inconsistent time series of positions.

I.3.1 Computation of the calibrated acceleration

Given the initial speed $v_0 = v(t_0)$ the velocity $v(t)$ is given by integration of the acceleration:

$$\begin{aligned} v(s) &= v(t_0) + \int_{t_0}^{s} \alpha(r)dr \\ &= v(t_0) + \int_{t_0}^{s} a(r)dr + \epsilon(s - t_0) \end{aligned} \tag{I.2}$$

Similarly the position $x(t)$ is given by:

$$\begin{aligned} x(t) &= x(t_0) + \int_{t_0}^{t} v(s)ds \\ &= x(t_0) + v(t_0)[t - t_0] + \\ &\quad \int_{t_0}^{t}[\int_{t_0}^{s} a(r)dr]ds + \epsilon(t - t_0)^2/2. \end{aligned} \tag{I.3}$$

The numerical integrations in Eq. (I.2) and Eq. (I.3) can be approximated by using the trapezoidal rule with timestep equal to the sampling interval.

In order to calculate the velocity and position from Eq. (I.2) and Eq. (I.3), we need to find an estimate for v_0 and ϵ. The QAR of the aircrafts measures the ground speed, from which the initial velocity could be gathered from. As these measurements sometimes give curves that are not smooth, they are not considered to be a very reliable source for the initial velocity. The very first measurements of the ground speed, which would be the initial velocity, are in addition extra unreliable, as it might takes a few seconds before the wheels spin properly after touchdown.

Due to these issues, it may be better to estimate both the bias term and the initial speed from the measured accelerations and GPS positions. These measurements are combined in a linear regression. The time points where we have recorded position values, are denoted by $t_1, ..., t_n$. Thus, we improve the precision by using all position values. We introduce the following notation for describing differences in time and positions:

$$\begin{aligned} s_i &= t_i - t_0, \quad i = 1, ..., n, \\ y_i &= x(t_i) - x(t_0), \quad i = 1, ..., n \\ z_i &= \int_{t_0}^{t_i}[\int_{t_0}^{s} a(r)dr]ds, \quad i = 1, ..., n. \end{aligned}$$

The relation between velocities, accelerations and positions from Eq. (I.3) can then be expressed as:

$$y_i - z_i = v_0 s_i + \epsilon s_i^2/2, \quad i = 1, ..., n. \tag{I.4}$$

By introducing the following matrix:

$$S = \begin{bmatrix} s_1 & s_1^2/2 \\ s_2 & s_2^2/2 \\ \vdots & \vdots \\ s_n & s_n^2/2 \end{bmatrix}$$

we can rewrite Eq. (I.4) as the following regression model:

$$(\boldsymbol{y} - \boldsymbol{z}) = S \begin{bmatrix} v_0 \\ \epsilon \end{bmatrix}, \tag{I.5}$$

where $\boldsymbol{y} = (y_1, ..., y_n)^T$ and $\boldsymbol{z} = (z_1, ..., z_n)^T$. The least-squares estimates for the unknown quantities v_0 and ϵ are given by:

$$\begin{bmatrix} \hat{v_0} \\ \hat{\epsilon} \end{bmatrix} = (S^T S)^{-1} S^T (\boldsymbol{y} - \boldsymbol{z}). \tag{I.6}$$

which gives us the parameters needed to calculate a calibrated acceleration. When using the calibrated acceleration $\alpha(t)$ to estimate the friction coefficient instead of the uncalibrated one $a(t)$, we refer to it as the calibrated friction coefficient.

I.3.2 The effect of calibration on the friction coefficient

To investigate the effect of calibration on the friction coefficient, the friction coefficients of both calibrated measurements μ_{cal} and the uncalibrated, raw measurements μ_{raw} are compared. A scatter plot between the two coefficients are given in Figure I.2, showing a Pearson correlation of 0.87.

In this paper, we use relative difference when comparing two numbers (a and b), defined by:

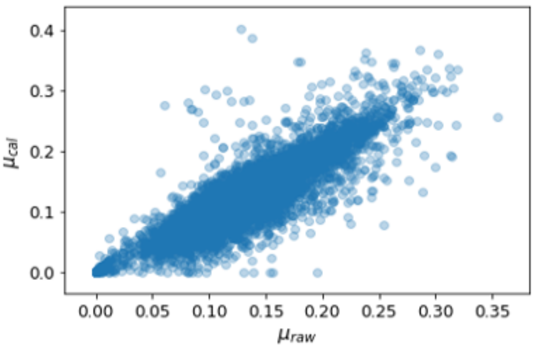


Figure I.2: Scatter plot between the calibrated and uncalibrated friction coefficients.

Table I.2: Summary for the calibrated and uncalibrated friction coefficient. *FricLim landings* refers the percentage of landings that are classified as friction limited.

Property	μ_{cal}	μ_{raw}	Diff_{rel}
Mean	0.121	0.127	-5.2%
Standard deviation	0.045	0.039	14.2%
FricLim landings	5.63%	5.38%	4.5%

$$\text{Diff}_{\text{rel}}(\text{a}, \text{b}) = \frac{\text{a} - \text{b}}{|\text{a} + \text{b}|/2} \tag{I.7}$$

unless otherwise is stated. A summary and the relative difference of the two coefficients are listed in Table I.2. The calibrated friction coefficient has a lower mean value, being 5.2% lower than the uncalibrated friction coefficient. Consequently, this increases the number of landings being classified as friction limited, such that the calibrated frictions lead to 4.5% more friction limited landings then the uncalibrated ones. In addition, the calibrated frictions are more scattered, with a standard deviation 14.2% higher than the uncalibrated. The distributions of both friction coefficients are shown in Figure I.3, and it can be seen how the calibrated friction coefficients tends to be a bit lower and more scattered than the uncalibrated coefficients.

The reasons why the calibrated friction coefficient tends to be lower can be revealed by getting a closer look at the relationships between the estimated initial speed v_0, the bias term ϵ and the difference between the calibrated and uncalibrated friction coefficients, which is denoted by μ_{diff} where $\mu_{\text{diff}} = \mu_{\text{cal}} - \mu_{\text{raw}}$. Figure I.4 shows a scatter plot between ϵ and μ_{diff}. Here we can see a strong linear relationship, with a Pearson correlation of $\rho = -0.97$. We see that when the bias term is positive, the calibrated friction is lower than the uncalibrated, and when the bias term is negative, the calibrated friction is higher

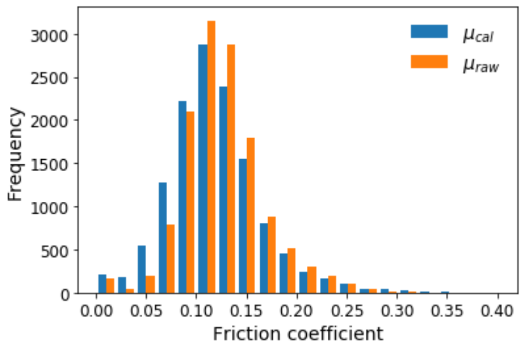


Figure I.3: Distributions of calibrated and uncalibrated friction coefficients.

than the uncalibrated. Around $e_p = 0$, μ_{diff} is also approximately zero, revealing that the bias term e_p is the main factor that distinguishes the calibrated value from the uncalibrated.

Going back to Eq. (I.1), we can see the reason behind the behavior of the calibrated friction. As the acceleration is a *retardation*, both $\alpha(t)$ and $a(t)$ are negative numbers. With a positive bias term, the calibrated value is closer to zero, which means that the retardation is reduced. A smaller retardation would in general mean less available friction, which again comes from a smaller friction coefficient between the tire and pavement. Thus, a positive bias term makes the calibrated friction coefficient smaller than the uncalibrated one, and the other way around for negative bias terms. As the bias terms has a positive mean value (+0.06 m/s), the calibrated friction coefficients are at average lower than the uncalibrated ones.

The initial speed v_0 has a smaller impact on the difference, as this is only a supporting variable estimated to calculate the bias term, and it is not directly used in calculating the friction coefficient. We still see a small tendency that the calibrated friction gets larger than the uncalibrated one when the initial speed increases, with a Pearson correlation between v_0 and μ_{diff} of $\rho = 0.42$.

We verify that the estimation of the initial speed from the data seems reasonable by comparing the difference between the measured initial velocity from ground speed v_0^g and the initial velocity estimated from the regression v_0. A summary of the differences is shown in Table I.3, and a scatter plot between the two initial velocities is given in Figure I.5. We see that the estimated v_0 is at average 4.9 m/s lower than the measured one, a relative difference of 7.5%.

I.3.3 Comparison with the SNOWTAM Reports

The friction coefficient is converted to Braking Action (BA), which is the international format for braking action declarations for SNOWTAM Reports, given by the International Civil Aviation Organization (ICAO). This is an integer in the range from 0 to 5, which the airport inspectors use to report the runway

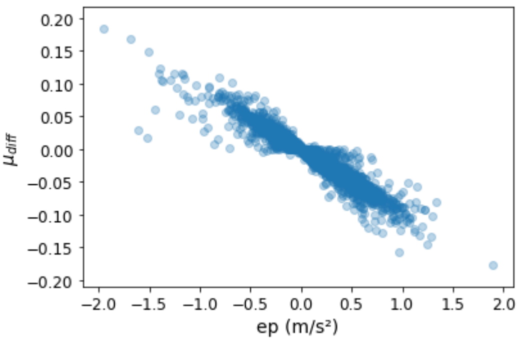


Figure I.4: Scatter plot of the bias term against the difference between the calibrated friction coefficients and the uncalibrated.

Table I.3: Summary of differences between v_0 estimated using linear regression and the measured v_0^g.

Property	Value
Mean v_0	62.6 m/s
Mean v_0^g	67.5 m/s
Difference in mean	-4.9 m/s
Relative difference in mean	-7.5%
Correlation	0.66
Mean absolute difference	5.7 m/s

Table I.4: Intervals for converting friction coefficients to the categorized braking actions.

Braking Action	Description	Friction Coefficient μ_B
0	NIL	[0.000, 0.050]
1	Poor	(0.050, 0.075]
2	Poor-medium	(0.075, 0.100]
3	Medium	(0.100, 0.150]
4	Medium-good	(0.150, 0.200]
5	Good	(0.200, ·]

surface conditions. The relationship between the friction coefficients and the braking action is given in Table II.1. This paper uses the same thresholds as Klein-Paste et al. 2012, which is based on aircrafts' landing distances as a function of braking action.

As the estimated friction coefficient will be used to evaluate and report the

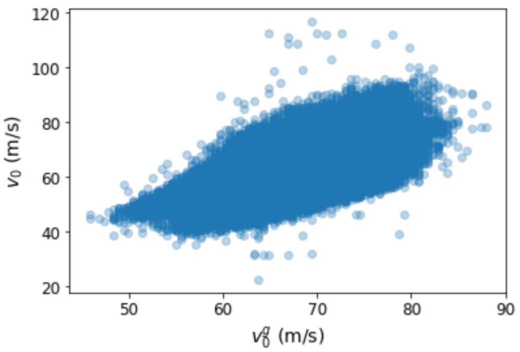


Figure I.5: Scatter plot of the estimated initial velocity v_0 against the measured initial ground speed v_0^g.

Table I.5: Percentage of BA which is equal or different between calibrated, uncalibrated and SNOWTAM BA (respectively *Cal*, *Raw* and *Snow*). The difference of +3, +4 and +5 are excluded, as they were all approximately zero.

Difference	Cal - raw	Snow - cal	Snow - raw
+2	0%	2%	3%
+1	8%	9%	10%
0	67%	34%	41%
-1	21%	29%	30%
-2	3%	17%	13%
-3	1%	6%	3%
-4	0%	3%	0%
-5	0%	1%	0%

Table I.6: Summary for the calibrated and uncalibrated friction coefficient converted to BA.

Property	Calibrated	Uncalibrated	Diff_{rel}
Mean	2.78	2.98	-6.8%
Standard deviation	1.07	0.89	20.6%
Mean absolute error	1.07	0.83	26.0%

quality of the SNOWTAM reports, it is important that the estimate is as accurate as possible. Therefore, we want to investigate the effect of the calibration with comparison with the SNOWTAM reports.

As SNOWTAM reports describe the runway braking action (BA) on a five-level scale, it is of interest to know how often the calibration makes the estimated runway braking action move from one level to another, as this will have a large effect on the comparison with the SNOWTAM reports. From the column *Cal - raw* in Table I.5 it can be seen that two thirds of the time, the calibrated and uncalibrated BA are the same. This means that one third of the time, the changes made when calibrating the friction coefficients are large enough for the BA to change one or more levels. Table I.6 shows the mean value and standard deviations of calibrated and uncalibrated BA, which gives a relative difference in mean of -6.8%.

When comparing the calibrated and uncalibrated BA with the SNOWTAM reports, we use the mean absolute error between the calculated BA and the SNOWTAM BA, which is shown in the last row of Table I.6. The mean error in SNOWTAM BA is 0.83 levels relative to the uncalibrated BA, and the mean error is 1.07 levels relative to the calibrated BA. We see that using the calibrated BA will increase the error of the SNOWTAM reports given under difficult runway conditions by 26%. This means that they will be considered as less accurate than they would if the uncalibrated BA was considered the true braking action. This

Table I.7: Summary for the calibrated friction coefficient using the estimated v_0 vs the measured v_0^g.

Property	μ_{v_0}	$\mu_{v_0^g}$	Diff$_{\text{rel}}$
Mean	0.121	0.143	-16.8%
Standard deviation	0.0445	0.0445	0.02%
FricLim landings	5.63%	5.06%	10.7%

is a relatively large difference and will have a significant impact on the evaluation of the accuracy of the SNOWTAM reports. The specific distributions of the error between the SNOWTAM reports and the calibrated and uncalibrated BAs are shown in column two and three in Table I.5.

I.3.4 Calibration using the measured initial speed

We consider another method for calibrating the acceleration, namely using the measured initial velocity and only estimating the bias term ϵ, even though the measurements seem unreliable. When the initial speed does not need to be estimated together with the bias term, the regression formula changes as described below.

If we denote the two columns in the matrix (I.6) by respectively:

$$\boldsymbol{s_1} = (s_1, ..., s_n)^T$$
$$\boldsymbol{s_2} = (s_1^2/2, ..., s_n^2/2)^T$$

we can rewrite Eq. (I.4) as the following regression model:

$$\hat{\epsilon} = (\boldsymbol{s_2}^T \boldsymbol{s_2})^{-1} \boldsymbol{s_2}^T (\boldsymbol{y} - \boldsymbol{z} - v_0^g \boldsymbol{s_1}) \tag{I.8}$$

which uses the measured initial ground speed v_0^g. A comparison of the results of the two calibration methods is shown in Table I.7. We see that using the measured initial speed had quite a large effect on the friction coefficient, having a higher mean value of 17% and consequently 11% less friction limited landings. $\mu_{v_0^g}$ also has a mean value larger than μ_{raw} given in Table I.2, which means that the two calibration methods have quite different effects on the friction coefficient, since μ_{v_0} has a mean value lower than μ_{raw}. This relatively large difference comes from the difference in the initial velocity (4.9 m/s) shown in Table I.3.

Until now, μ_{v_0} has been considered the most appropriate friction coefficient based on experience with the measurements, and it is used in the warnings systems of IRIS. As there is no measured ground truth to compare the friction coefficients with, it is not a simple task to show which method gives the most accurate values. This will be subject to further exploration, which will include comparisons with weather data.

I.4 Stability of brake pressure

The measurements of brake pressure have a high influence on several aspects of the calculations. It has a main contribution on deciding whether a landing is friction limited or not, but it also has an important contribution to calculating the friction coefficient during the landing. We have therefore done an exploration of the stability of the brake pressure, and whether all measurements along the time series of the landings are reliable, or if some parts of the landing should be filtrated out. Especially, we have investigated the different parts of the landing according to the use of thrust reversal, a diversion of the aircraft's engine thrust such that it contributes to the deceleration during landing. It is of interest to verify that the brake pressure is usable during the phase when the thrust reversal is deployed, meaning confirming that the pilots apply the brakes to a high degree also when the thrust reversal system is in use. There has been some uncertainty to how the measurements of the brake pressure are affected by the use of thrust reversal systems, and especially if the measurements are unstable in the period where the thrust reversal is turned on/off, also called the transit phase. Therefore, we have done a specific exploration of the stability of the measurements during the transit phase.

The thrust reversal system is almost always in use when landing on Nordic airports, as it contributes to shorter landing distances and reduces wear on the brakes. On slippery runways, the use of thrust reversal is of major importance to avoid accidents. The thrust reversal is eventually turned off when the aircraft's speed has slowed down. This is because of the high decrease in the effect of the thrust reversal when the aircraft's speed is low, and that using it on a slower moving vehicle could inflict damage as a consequence of pushing debris into the engine, as described in Oda et al. 2010.

To investigate the relevance of the different phases for calculating the friction coefficient, we look at the normal development of brake pressure during a landing, where a typical example is given in Figure I.6. We find a tendency for the highest brake pressures to occur during the phase when the thrust reversal system is in use, as seen in Table I.8. The column *% of total* shows the distribution of the total occurrences of high pressure, distributed on the three different phases. We see that approximately 75% of the high pressure values occur during the thrush reversal phase, as well as 72% of the extreme values, meaning this is the phase which affects the calculation of the friction coefficients the most. The column *% of time* shows the percentage of time each phase has high brake pressures. We find that 7.9% of the time in the phase with thrust reversal, we have a brake pressure higher than 1000 psi. This happens for only 1.5% of the time with no thrust reversal used.

It would seem that the transit phase does not have larger variations or more extreme values in the brake pressure, as only 5.1% of the brake pressures in this phase is above 1000 psi, a lower number than for the thrust reversal phase. In addition, this phase contributes to only 11% of the high pressure values and 8.1% of the extreme values. As the calculations of the friction coefficients depends mostly on the high pressure values, it would seem that this phase does not have

Table I.8: Summary of occurrence of high brake pressure, giving the distribution of the total occurrences of high pressure, and amount of time each phase has high pressure.

Phase	% of total	% of time
Bpr $\geq$ 1000 psi		
No thrust reversal	15.1%	1.5%
Thrust reversal	74.6%	7.9%
In transit	10.3%	5.1%
Bpr $\geq$ 2000 psi		
No thrust reversal	20.0%	0.4%
Thrust reversal	71.8%	1.7%
In transit	8.1%	0.9%

the highest contributions to the calculations. Nevertheless, as the stability of these measurements has been questioned for some time, we verify this result by investigating the effect of filtrating out the transit phase, to see how large impact this has on calculating the friction coefficient. This is done by calculating the friction coefficient for four different kinds of filtration:

- No filtration
- Filtration of the transit phase only
- Filtration of the transit phase and one second on each side
- Filtration of the transit phase and two seconds on each side

The result of this is shown in Table I.9, where it can be seen that there is not much difference in the friction coefficients' mean for the different kind of filtrations, where the largest absolute relative difference is 0.30%. The P-values

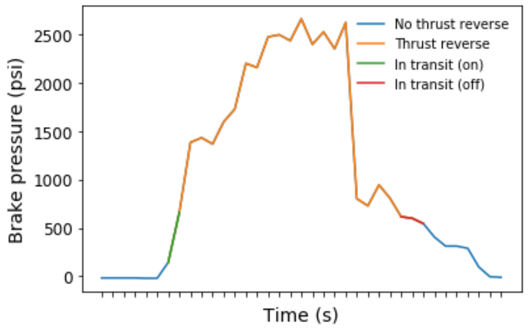


Figure I.6: A typical example of brake pressure after touchdown with color marking according to phase.

Table I.9: Summary of mean values for different kind of filtration. *Difference* is the relative difference between the mean of that filtration method and the mean of using no filtration, Difference = $(\mu_{\text{filter}} - \mu_{\text{none}})/\mu_{\text{none}}$. *P-value* is the P-value for the difference in mean in relation to using no filtration, and *FricLim* refers to the percentage of landings which are classified as friction limited.

Filtration	Mean	Difference	P-value	FricLim
None	0.1209	0%	1	5.63%
Only transit	0.1212	0.24%	0.602	5.58%
One second	0.1213	0.30%	0.520	5.52%
Two seconds	0.1207	-0.21%	0.654	5.37%

are quite large, giving no signs of significant differences in the means. The largest variation is seen in number of friction limited landings, which decreases from 5.63% to 5.37%. The results indicates that the transit phase does not have instability problems that affect the calculations of the friction coefficients, and that it is appropriate to use data from all phases of the landing in the calculations.

I.5 Conclusions and future work

In this paper we have discussed some of the challenges with converting time series of measurements of flight data into reliable estimates of runway friction. We have shown how to calibrate the acceleration measurements to account for bias in the measurements, both using the measured initial velocity and without using it. We have also shown the effect the calibration has on the estimated friction coefficient and the evaluation of the accuracy of SNOWTAM reports.

We found that the calibrated friction coefficient is at average lower and more scattered than the uncalibrated friction coefficient, which leads to an increase in the number of friction limited landings. Furthermore, the changes made on the friction coefficient has a significant impact on the evaluation of the accuracy of the SNOWTAM reports, as they will be considered less accurate than they would if the uncalibrated coefficients were considered the truth. In addition, we have shown the behavior and stability of the measurements of brake pressure during the flight landing in conjunction with using thrust reversal.

This work is a small part of a large study with the main goal of providing pilots and airport operators with accurate and timely information about the actual runway surface condition, to support in safe and economic operations of airports. In addition to verifying existing warning models, the estimates of runway frictions will be used as a response variable when using machine learning to predict the runway surface conditions, based on weather forecasts. Further work will also include exploring methods for evaluating the different approaches for estimating the friction coefficient.

Acknowledgements. The authors are grateful to Avinor for making the flight data and runway data available to this research project. We also want to thank Boeing for invaluable help regarding the use of flight data.

References

Giesman, P. (2005). “Wet runways, physics, certification, application”. In: *Boeing Performance and Flight Operations Engineering Conference*, no. 8, pp. 1–24.

Huseby, A. B., Klein-Paste, A., and Bugge, H. J. (2010). “Assessing airport runway conditions – A Bayesian approach”. In: *Reliability, Risk and Safety Back to the Future*. Ed. by B, A., I, P., and E, Z. London CRC Press, pp. 2024–2032.

Huseby, A. B. and Rabbe, M. (2008). “Predicting airport runway conditions based on weather data”. In: *Safety, Reliability and Risk Analysis: Theory, Methods and Applications*. Ed. by S, M., C, G. S., and J, B. London CRC Press, pp. 2199–2206.

— (2012). “A scenario based model for assessing runway conditions using weather data”. In: *11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference*. Ed. by PSAM and ESREL. Curran Associates, Inc, pp. 5092–5101.

— (2018). “Optimizing warnings for slippery runways based on weather data”. In: *Safety and Reliability - Safe Societies in a Changing World. Proc. ESREL 2018*.

Klein-Paste, A. et al. (2012). “Braking performance of commercial airplanes during operation on winter contaminated runways”. In: *Cold Regions Science and Technology*, no. 79–80, pp. 29–37.

Oda, H. et al. (2010). “Safe Winter Operations”. In: *AERO*, no. 4, pp. 5–14.

Rosenker, M. V. et al. (2007). “Runway Overrun and Collision Southwest Airlines Flight 1248”. In: *Aircraft Accident Report, National Transportation Safety Board*, no. NTSB/AAR-07/06 PB2007-910407.

Søderholm, B. et al. (2009). “Integrated Runway Information System (IRIS)”. In: *Project Report, Avinor, Norway*. In Norwegian.

Aarrestad, O. et al. (2007). “Safe Winter Operation Project (SWOP)”. In: *Project Report, Avinor, Norway*. In Norwegian.

Paper II

A decision support system for safer airplane landings: Predicting runway conditions using XGBoost and explainable AI

Alise Danielle Midtfjord, Riccardo De Bin, Arne Bang Huseby

Published in *Cold Regions Science and Technology*, 2022, volume 199, issue 103556, DOI: 10.1016/j.coldregions.2022.103556

II

Abstract

The presence of snow and ice on runway surfaces reduces the available tire-pavement friction needed for retardation and directional control and causes potential economic and safety threats for the aviation industry during the winter seasons. To activate appropriate safety procedures, pilots need accurate and timely information on the actual runway surface conditions. In this study, XGBoost is used to create a combined runway assessment system, which includes a classification model to identify slippery conditions and a regression model to predict the level of slipperiness. The models are trained on weather data and runway reports. The runway surface conditions are represented by the tire-pavement friction coefficient, which is estimated from flight sensor data from landing aircrafts. The XGBoost models are combined with SHAP approximations to provide a reliable decision support system for airport operators, which can contribute to safer and more economic operations of airport runways. To evaluate the performance of the prediction models, they are compared to several state-of-the-art runway assessment methods. The XGBoost models identify slippery runway conditions with a ROC AUC of 0.95, predict the friction coefficient with a MAE of 0.0254, and outperforms all the previous methods. The results show the strong abilities of machine learning methods to model complex, physical phenomena with a good accuracy.

II.1 Introduction

Contamination of runway surfaces with snow, ice or slush causes potential economic and safety threats for the aviation industry during the winter seasons. The presence of these materials reduces the available tire-pavement friction needed for retardation and directional control, which can lead to accidents and loss of human lives (Giesman 2005; Klein-Paste, Huseby, et al. 2012). During 2019, seven commercial passenger aircrafts ran out of the runways during landing in the United States due to bad weather and runway conditions (Foundation n.d.). Difficult landing conditions is not only a problem at northern airports. On 7th August 2020, an aircraft suffered a runway excursion in India during poor weather conditions, and both pilots and 19 passengers died in the accident. Difficult weather conditions such as snow and rain also contribute to the increasing growth of delayed and cancelled flights (X. Zhang and Mahadevan 2017).

If the aviation industry returns to the growth trajectory it had before COVID-19, the global air transport demand is expected to triple within the year 2050 (Gőssling and Humpe 2020), which will increase the need for more efficient and safer operations of airports runways. The increase in extreme weather conditions due to the climate change also rises problems for aviation operations both in the air and on the ground (Coffel and Horton 2015; Gultepe et al. 2019). This has led the global aviation industry to work towards more standardized and data-driven assessment of runway conditions (Kornstaedt 2021), which pilots need to activate appropriate safety procedures when landing and at take-off. Information about the available friction on the runways are given to pilots in international standardized runway reports. Unfortunately, the accuracy of runway reports can sometimes be unsatisfactory, and measuring the runway friction with an acceptable precision is difficult (Anupam et al. 2017; Niu et al. 2020). While many different friction measurement devices have been developed, it is hard to find equipment that produces stable and consistent results which corresponds to the experienced braking friction for landing aircrafts (V. V. Putov, A. V. Putov, Sheludko, et al. 2015; V. V. Putov, A. V. Putov, Kazakov, et al. 2015). Another problem is that in order to measure friction, the runway must be closed for traffic. Thus, such measurements cannot be carried out too frequently. Especially heavy snowfalls or sudden drops in temperature may result in rapidly changing conditions. As a result, the runway reports are not as useful as one could hope.

There have been several studies that attempt to make the measurements from the friction measurement devices more useful. They relate the ground friction measurements to the aircraft braking friction using correlations or adjustments (Cerezo et al. 2016; Joshi et al. 2015; Rado and Wambold 2014). However, the inconsistency and variance between friction measurement devices and different airports is a problem. Acoustic (Alonso et al. 2014; Kongrattanaprasert et al. 2009), tread (Erdogan, Alexander, and Rajamani 2011; Niskanen and A. J. Tuononen 2014) and optical (Holzmann et al. 2006; A. Tuononen 2008) measurements have also been considered (for a review, see Khaleghian, Emami, and Taheri 2017). In the development of better anti-skid brake systems, there have been studies on using sensor data of landing aircrafts, such as wheel speed and brake force, to provide real-time estimation of the available braking friction force (Jiao, Sun, et al. 2019; Jiao, Z. Wang, et al. 2021; C. Lee, Hedrick, and Yi 2004).

One problem with the measurement methods is that they depend on real-time

Table II.1: Description of braking action together with intervals for converting friction coefficients to this format.

Braking Action	Description	μ_B
0	NIL	[0.000, 0.050]
1	Poor	(0.050, 0.075]
2	Poor-medium	(0.075, 0.100]
3	Medium	(0.100, 0.150]
4	Medium-good	(0.150, 0.200]
5	Good	(0.200, ·]

measurements from sensors attached to the aircraft, which measure the relevant parameters as the vehicle challenges the friction. Pilots need to know the surface conditions prior to landing, meaning these methods are not useful in our case. To address this issue, there have been conducted some studies on how available surface friction is affected by weather conditions and runway contamination mainly based on engineering- and physics-based models and basic statistical approaches. Klein-Paste et al. (2015) Klein-Paste, Bugge, and Huseby 2015 proposed a runway model for the surface conditions which interprets descriptive data from the international standardized runway reports called Snowtam reports. The model evaluates a sum of seven effects that contain information about the runway contamination as well as measurement of runway temperature and humidity, $P = \sum_{i=1}^{7} x_i$. The first effect x_1 sets the main assessment of the runway conditions in the interval $[1, 5]$ by evaluating the form of contamination on the runway. Then, the assessment is either upgraded or downgraded by considering the next six factors, which have values in the range of $[-2, 2]$. This includes the effect of spatial coverage x_2, the depth of contamination x_3, runway temperature x_4, humidity x_5, and the use of chemicals x_6 and sanding x_7. The output of the model is a prediction P of the runway *braking action*, which is the international format for specifying runway conditions and is described in Table II.1. When P exceeds 5 it is set to 5, and when it is lower than 1 it is set to 1.

W. Zhang et al. 2021 recently performed a quantitative analysis of the relationship between braking performance and different factors such as runway treatment and slope, precipitation, and contamination type, by using data from airports in the United States. The work gives explorative insights into the relationships between these effects and braking performance, but does not provide a model which predict the surface conditions for new airplane landings.

Juga, Nurmi, and Hippi 2013 predicted surface friction on traffic roads using linear regression models with weather data, which can be partially related to the surface friction on airport runways. The models use the road surface temperature and thickness of contamination as input and predict the friction coefficients,

$$
\begin{aligned}
\mathrm{CF}_{si} &= a_1 f(X_S) + b_1 f(X_I) + c_1 f(T_r) + d_1 \\
\mathrm{CF}_w &= a_2 f(X_W) + d_2 \\
\mathrm{CF}_d &= 0.82
\end{aligned}
$$

where CF_{si}, CF_w and CF_d represent the friction coefficient for snowy/icy, wet and dry runways, T_r is the runway temperature, X_S, X_I and X_W are the thickness of snow, ice and water layers, and a_i, b_i, c_i and d_i ($i \in \{1, 2\}$) are the regression coefficients. The model is used in the road weather model *RoadSurf* in Finland, which simulates road surface temperatures, conditions and friction coefficients to assist in traffic safety and winter road maintenance (Kangas, Heikinheimo, and Hippi 2015). Another study on surface friction on traffic roads is done by S. Kim, J. Lee, and Yoon 2021, where the friction coefficient on roads is predicted during rainy weather. This is done with an artificial neural network using rainfall intensity, water film thickness, and road surface temperature as input variables and the tire-to-road friction coefficient as response.

Huseby and Rabbe 2012 introduced a scenario-based model for assessing airport runway conditions using weather data, which defines a set of scenarios known to cause slippery conditions. By monitoring the meteorological parameters runway temperature, air temperature, relative humidity, horizontal visibility, and precipitation type and intensity, the model detects slippery scenarios. As an example, the scenario $SNOW$ is one that can happen quite often at northern airports, and the precise mathematical conditions for this scenario are:

- $pt_i \in \{\text{snow, sleet, drifting snow}\}$ at least once, $i \in I_4$,
- $ta_i \in [-8°C, +2°C]$ for all $i \in I_4$,
- $tr_i \leq 0°C$ for all $i \in I_4$,
- $hu_i \in [85\%, 100\%]$ for all $i \in I_4$,

where pt_i is the precipitation type at time i, ta_i and tr_i are the air temperature and runway temperature, hu_i is the relative humidity, and I_4 is a time slot containing the last four hours from the given point of time. The scenario model was further developed in Huseby and Rabbe 2018, where it was shown that the scenario model could be improved by optimizing the thresholds for the scenarios according to Type 1 and Type 2 errors using weather and flight data. Both Huseby and Rabbe 2012's scenario model and the runway model of Klein-Paste, Bugge, and Huseby 2015 are used in an integrated runway information system called *IRIS*, which is implemented on 16 Norwegian airports to support safer operations of Norwegian airports.

The complexity and non-linearity of the physical relationships controlling the surface friction, and their dependency on each other through time, makes it difficult to provide precise physical models of the experienced runway surface friction for landing aircrafts. Machine learning have on several occasions shown to be able to model complex physical phenomena with a good accuracy, which has increased the focus on the use of this technology when predicting the behaviour of physical phenomena such as snow and ice (e.g. De Coste et al. 2021; Kellner et al. 2019; Montewka et al. 2015; Zhu et al. 2021). The main objective of this paper is to study if machine learning methods can predict runway surface conditions with a higher precision than previous methods, and contribute to safer airplane landings. This is done by using XGBoost to create a combined system of a classification model and a regression model trained on weather data, data from runway reports, and sensor data from landing aircrafts through an airplane performance model. Similar to other tree ensemble methods, tree-based XGBoost with deep decision trees does not provide a directly interpretable model. Therefore, we use SHAP to create simplified, understandable models that provide both global and local explanations of the models' predictions. All the models are combined to create a data-driven decision support system, which can aid airport operators and pilots in their decisions and contribute to safer and more economic operation of airports. To evaluate the performance of the XGBoost models, they are compared to state-of-the-art runway surface conditions assessment models, namely the *runway model* of Klein-Paste, Bugge, and Huseby 2015 and the *scenario model* ofHuseby and Rabbe 2012. The models are also compared to runway assessment of airport runway inspectors reported in the *Snowtam reports*.

The rest of the paper is structured as follows: In section 2 we briefly describe the data and sources used in this work, and how the response variable and explanatory variables are extracted. In section 3 we describe XGBoost and SHAP, which is the methodology used to create the models. In section 4 we evaluate the performance of the models and compare them the runway model, the scenario model and the assessment from runway inspectors. We also describe the XGBoost models by using SHAP values to create global explanations. In section 5 we introduce the decision support system, which combine the output from the XGBoost models together with local explanations of the predictions. In section 6 we sum up the work and add our conclusive remarks and future work. The implementations of the methods in Python, as well as the final trained XGBoost models, are available at https://github.com/alimid/surface_friction.

II.2 Data sources and variable extraction

II.2.1 Data sources

All data used in this study are made available by Avinor, the largest airport operator in Norway. The full data set includes weather data, runway reports and flight data from 13 Norwegian Airports. There are significant differences between these airports with respect to weather conditions, maintenance procedures, runway lengths, and traffic. To avoid possible effects of these differences on the analysis, separate models should be fitted for each airport. This study focuses on data from Oslo Airport, Norway's largest airport.

The weather data comes from measurement devices at the airport, which measures meteorological variables every minute such as wind speed, temperature, humidity, and precipitation.

The runway reports, called Snowtam reports, are created by the airport operators and include descriptive information about runway contamination such as type and depth and maintenance procedures such as sanding of the use of chemicals. A new Snowtam report must be issued at least every 24 hours or when the runway conditions change significantly.

The flight data, collected over ten winter seasons from season 2009/2010 to season 2018/2019, is provided by Scandinavian Airlines Service (SAS) and Norwegian Air Shuttle AS and is gathered from the Quick Access Recorder (QAR) of Boeing 737-600/700/800 NG airplanes. The flight data for Oslo Airport consists of 200 508 landings. The flight data is used to estimate the friction coefficient and calculate whether a landing is friction limited or not, as described in Section II.2.2. For each landing, the data consists of 60 seconds of measurements such as acceleration, brake pressure, flap position, and engine thrust starting from touch down.

II.2.2 Calculating the response variable

To reflect the airport runway conditions, the aircraft braking coefficient, μ_B, is calculated using a performance model developed by Boeing. μ_B is defined as the ratio of the tangential force needed to maintain uniform relative motion between the aircraft's wheels and the runway surface. The calculations are based on the equation of motion of a moving vehicle

$$m\frac{dv}{dt} = D_{\text{thrust}} - D_{\text{aero}} - mg\sin(\epsilon) - D_{\text{brakes}}, \tag{II.1}$$

where m is the weight of the aircraft, $\frac{dv}{dt}$ is the acceleration, D_{thrust} is the force caused by thrust, D_{aero} is the aerodynamic drag, g is the gravitation, ϵ is the slope of the runway, and D_{brakes} is the force contribution from the wheels. The contribution of D_{thrust}, D_{aero}, and D_{brakes} can be calculated using aircraft-type specific performance models, and μ_B can then be retrieved from D_{brakes}

$$\mu_B = \frac{D_{\text{brakes}}}{mg\cos(\epsilon) - L}, \tag{II.2}$$

where L is the aerodynamic lift. At Oslo airport, ϵ was set to zero as the contribution in retardation due to slope was negligible. As the friction coefficient is a dimensionless scalar dependent on the characteristics of the two touching bodies (the road and the aircraft wheels), the actual friction coefficient is unrelated to airplane type, and can be universally used for all airplane landings. More details about calculating the braking coefficient can be found in Midtfjord and Huseby 2020 and Klein-Paste, Huseby, et al. 2012.

An important problem when analysing flight data is deciding whether a landing is *friction limited* or not. Unless the pilot challenges the runway friction during the landing by fully applying the brakes, the maximum friction available will not be utilized. In this case, μ_B reflects the amount of tire-pavement friction that was used. When wheel brakes are applied fully or to a high degree on slippery runways, the maximum attainable friction from the runway is used during the stop. In this case, the aircraft's deceleration is limited by the friction available from the runway, and the obtained μ_B will reflect the amount of tire-pavement friction that is available.

To figure out if the brakes are applied fully during a landing, we check whether the brake pressure "requested" by the pilot exceeds the brake pressure estimated based on the measured deceleration. Whenever this occurs, the anti-skid system is activated, and all the available friction is used. If these conditions last for at least 3 successive seconds, the landing is said to be friction limited. Since the braking coefficient then reflects the available tire-pavement friction, we refer to it as the *friction coefficient*.

In the first part of our system, we want to classify whether a landing is slippery or not, i.e. we want to get a warning when the runway conditions are not good. If a landing is friction limited, this indicates that the runway conditions may not be optimal. However, this does not necessary imply that the runway conditions are bad. The friction coefficients can be converted to the corresponding braking actions by using the international standardized values in Table II.1. Landings which are friction limited and have a friction coefficient $\mu_B \leq 0.15$, are classified as slippery, as this corresponds to a medium or worse braking action. Landings which are friction limited and have a friction coefficient $\mu_B > 0.15$, are classified as non-slippery, as this corresponds to a braking action which is medium-good or good. In order to simplify the terminology, landings which are non-friction limited are also classified as non-slippery. It should be noted, however, that most likely several of the non-friction limited landings may have been subject to slippery conditions as well. However, due to limitations in the method used for estimating the friction coefficient, it is not possible to identify these landings with a satisfactory level of certainty. Nevertheless, this simplification of terminology allows us to use information from all airplane landings, also the ones where the friction coefficient is unknown due to the landing being non-friction limited. It should be pointed out that when a landing is non-friction limited, this is an indication of non-slippery conditions, as there is likely a positive correlation between whether a landing is friction limited and whether the conditions are slippery. At least, the runway conditions can not be "dangerously" slippery for non-friction limited landings, since the airplane does not even need to use all available friction. Table II.2 shows the distribution of the landings at Oslo Airport at the winter seasons 2009/2010 until 2018/2019, where the landings are classified as *slippery* for 5 163 of the 200 508 landings, which is only 2.57% of the landings.

In the second part of the system, we want to model *how* slippery the conditions are, by only considering the observations for which we can actually estimate the true runway friction. This is done by using an XGBoost regression algorithm on the friction coefficients for the friction limited landings. The predicted friction coefficients are then converted to braking actions according to Table II.1, to comply with international standards.

Table II.2: Number of landings at Oslo Airport in our dataset for the winter seasons 2009/2010 until 2018/2019.

Class	Description	Number of landings
Non-slippery	Non-friction limited	193 056
	Friction limited and $\mu_B > 0.15$	2 289
Slippery	Friction limited and $\mu_B \leq 0.15$	5 163

II.2.3 Extracting the explanatory variables

The effect weather has on the runway surface conditions is complex as it is highly dependent on the interaction between multiple weather variables over time, as well as the maintenance of the runways. It is not enough to simply consider the present weather; it is also necessary to know how the weather has been backwards in time and what kind of maintenance operations has been carried out on the runway in the meantime.

One way to include both some information about maintenance operations on the runway as well as weather development some time backwards from the present is to include data from the Snowtam reports in the variables. The reports include information about the maintenance actions sanding and the use of de-icing or anti-icing chemicals on the runways. The reports also contain information about runway contamination such as snow, rime, or ice, as well as the depth and coverage of the contamination. By using the reports, it is possible to gather knowledge about past precipitation and temperature development. However, since the Snowtam reports are issued only one to a few times per day, they do not provide information about rapid changes. Therefore, real-time information about weather development backward in time should be drawn from measurements of meteorological variables in addition to the data from the Snowtam reports. One commonly used method of capturing relationships between time series of multiple variables, is to include time lags (past measurements) of the variables as new explanatory variables, which is commonly done in e.g. the statistical Vector Autoregression (VAR) models. Since VAR models assume linear relationships between the present variable value and variables' time lags, we generalize this framework such that the effect of the variables' time lags on the response can be any function (e.g. decision trees):

$$y_t = f(X_{t-p}, X_{t-p+1}, \cdots, X_{t-1}) \tag{II.3}$$

where y_t is the friction coefficient at time t, and X_{t-p} is the matrix of explanatory variables at time step p backwards from t. In this work, the time lags of the following variables are included:

- pt = Precipitation type

- pi = Precipitation intensity
- ta = Air temperature
- tr = Runway temperature
- hu = Relative humidity
- vi = Horizontal visibility
- ap = Air pressure
- dp = Dew point

where the resolution is one measurement per minute. To capture the evolution of the explanatory variables over the relevant time span, without increasing the dimensionality of the variable matrix too much, it was decided to include time lags of $k \in \{1, 3, 6, 12, 24\}$ hours back in time. Adjusting the notation for the minute-hour codification, we consider

$$x_{i,k} = x_{i-60 \cdot k} \tag{II.4}$$

where $x_{i,k}$ denotes variable x at k hours backwards from time i and $x \in \{pi, ta, tr, hu, vi, ap, dp\}$. These time lags and variables were chosen according to expert knowledge of runway friction and meteorology. Using a similar notation, we also include the trend of some relevant variables over time, by taking the difference between the present value and their values k hours back in time:

$$\Delta_k x_i = x_i - x_{i-60 \cdot k}, \tag{II.5}$$

where $x \in \{tr, hu, ap\}$. These variables were chosen as their trend might affect surface conditions, especially when large changes occur. In addition, precipitation over time was included by accumulating their intensity:

$$ac\_pt_{i,k} = \sum_{j=i-60 \cdot k}^{i} pi_j \cdot I_{\{pt_j = pt_i\}}, \tag{II.6}$$

where $pt \in$ {rain, sleet, wet snow, dry snow} and $I_{\{pt_j = pt_i\}}$ is the subset where the precipitation type is of the type pt_i between times k and i. In addition to the mentioned variables, present measurements of along wind and across wind were also included in the explanatory variables. We have also included the absolute value of air temperature and runway temperature, as temperatures closer to zero can lead to difficult runway conditions, independent of the sign.

Another challenge when working with weather data and runway reports are the categorical variables. Especially the contamination type in the Snowtam reports has a complex setup; it consists of nine different contamination codes given in Table II.3, where the final category can be a combination of several layers. As an example, the contamination code 479 means *Dry snow* on *ice* on *Frozen ruts or ridges*. The multiple layers consist of maximum one "loose layer" and maximum two "solid layers". One way to make these combinations more useful is to create groups of contamination codes. Klein-Paste, Bugge, and

Table II.3: Contamination codes and types reported in the Snowtam reports.

Code	Description
0	Bare and Dry
1	Damp
2	Wet
3	Rime
4	Dry Snow
5	Wet Snow
6	Slush
7	Ice
8	Compacted or rolled snow
9	Frozen ruts or ridges

Huseby 2015 divided the different combinations of contamination codes into six groups based on their slippery characteristics, and used the groups in further calculations in the runway model:

- Not contaminated
- Dry contaminated
- Wet Contaminated
- Solid Contaminated
- Loose and dry Contaminated
- Solid base layer

One combination of contamination codes can occur in several of the groups. Another way to decrease the number of possible combinations is to narrow down to report only two layers, which is the future approach the international format for specifying runway conditions is going to take (Rodriguez 2019).

One benefit of XGBoost, which is the machine learning algorithm used to train the runway surface condition predictor in this work, is that it handles sparse data well, as it uses a sparsity-aware split finding algorithm (T. Chen and Guestrin 2016). Therefore, it is possible to enter the complex, categorical variable *contamination type* as several one-hot encoded variables, one for each possible combination of contamination codes. One-hot encoding is a transformation of the original variable with N possible states to N binary variables, one for each possible state. The variable *contamination type* was transformed to 30 one-hot encoded variables. The same one-hot encoding was done for the weather variable *precipitation type*, which is also a categorical variable with nine categories. The final matrix with explanatory variables consisted of 151 variables, which are shown in Appendix II.A.

II.3 Methodology

II.3.1 eXtreme Gradient Boosting

In this paper, we build prediction models using the state-of-the-art boosting algorithm XGBoost (T. Chen and Guestrin 2016), to predict runway conditions using weather data and data from runway reports as input variables. XGBoost stands for eXtreme Gradient Boosting and is a scalable implementation of gradient boosting decision trees (Friedman 2001). Since its release in 2014, XGBoost has been a very popular machine learning method, and it has a highly impressive winning record when it comes to machine learning competitions. XGBoost has already been used in several transportation risk assessment applications both within road traffic (Parsa et al. 2020; Shen and Wei 2020; Shi et al. 2019), aviation (Li et al. 2020), and shipping (Adland et al. 2021; He, Hao, and X. Wang 2021; Jin et al. 2019).

XGBoost is a supervised learning method, so it derives a model $f(\boldsymbol{x})$ that relates m input variables $\boldsymbol{x}$ to an outcome of interest y. This is done by minimizing a loss function $L(y, f(\boldsymbol{x}))$ that penalizes differences between y and $f(\boldsymbol{x})$. As a boosting approach, XGBoost does not minimize the loss function at once, but in small steps. This is done by iteratively fitting a weak learner, in this case a penalised version of a decision tree, to the gradient of the loss computed at the previous iteration. The final model estimate $\hat{f}(\boldsymbol{x})$ will have the form

$$\hat{f}(\boldsymbol{x}) = \sum_{k=1}^{K} f_k(\boldsymbol{x}), \tag{II.7}$$

where $f_k(\boldsymbol{x})$ is the decision tree computed at iteration k. In contrast to other ensemble methods like bagging and random forests, a boosting algorithm learns from the results of the previous iteration. In this way, the algorithm can focus on the most interesting data structures, and the space of the possible models is better explored.

In practice, the model must be learned from the data, which in general consist of a n (number of observations) times m (number of variables) matrix of input X and a n-dimensional vector of outcomes $\boldsymbol{y}$. At each iteration, a decision tree $f_k(\boldsymbol{x})$ is derived by minimizing an objective function

$$\text{obj}(f_k(\boldsymbol{x})) = \sum_{i=1}^{n} L\left(y_i, \hat{f}(\boldsymbol{x}_i)^{[k-1]} + f_k(\boldsymbol{x}_i)\right) + \Omega(f_k(\boldsymbol{x})) \tag{II.8}$$

where $(\boldsymbol{x}_i, y_i)$ is the i-th observation, $\sum_{i=1}^{n} L\left(y_i, \hat{f}(\boldsymbol{x}_i)^{[k-1]} + f_k(\boldsymbol{x}_i)\right)$ is the empirical estimate of the loss, $\hat{f}(\boldsymbol{x}_i)^{[k-1]}$ is the current estimate of the model (i.e., the model computed at the previous iteration $k-1$), and $\Omega(f_k(\boldsymbol{x}))$ is a penalty term that penalizes the tree complexity.

Basically, at iteration k, XGBoost looks for the tree $f_k(\boldsymbol{x})$ that better improves the current model $\hat{f}(\boldsymbol{x})^{[k-1]}$. Due to the boosting requirement of a weak learner, the optimization is constrained by $\Omega(f_k(\boldsymbol{x}))$, such that simple trees are favoured. Once the best tree $f_k(\boldsymbol{x})$ is obtained, its contribution is added to the current model,

$$\hat{f}(\boldsymbol{x})^{[k]} = \hat{f}(\boldsymbol{x})^{[k-1]} + \nu f_k(\boldsymbol{x}). \tag{II.9}$$

Note that the k-th contribution to the final model is actually shrunk by a factor ν (step size shrinkage), which reduced the convergence speed and therefore fulfils the boosting requirement of making only a small improvement to the model at each iteration.

Part of the success of XGBoost lies in its clever way to perform the optimization above. Instead of working directly with Eq. (II.8), the optimization is performed on its second order approximation

$$\text{obj}(f_k(\boldsymbol{x})) \approx \sum_{i=1}^{n} \left[L\left(y_i, \hat{f}(\boldsymbol{x}_i)^{[k-1]}\right) + g_i f_k(\boldsymbol{x}_i) \ + \tfrac{1}{2} h_i f_k^2(\boldsymbol{x}_i) \right] + \Omega(f_k(\boldsymbol{x})), \tag{II.10}$$

where

$$\begin{aligned} g_i &= \partial_{\hat{f}(\boldsymbol{x}_i)^{[k-1]}} L(y_i, \hat{f}(\boldsymbol{x}_i)^{[k-1]}) \\ h_i &= \partial^2_{\hat{f}(\boldsymbol{x}_i)^{[k-1]}} L(y_i, \hat{f}(\boldsymbol{x}_i)^{[k-1]}). \end{aligned}$$

The key point is that the construction of the decision trees, namely the identification of the split points and the leaf weights, only depends on the loss function through these two gradient terms, which makes the computations easier. The formulation of $\Omega(f_k(\boldsymbol{x}))$, calculated as

$$\Omega(f_k(\boldsymbol{x})) = \gamma T_k + \frac{1}{2}\lambda ||w_k||^2 \tag{II.11}$$

also helps the computations, as it associates a penalty parameter γ to the computation of the split points and a penalty parameter λ to that of the leaf weights. The former parameter penalizes the number of tree leafs T, the latter the magnitude of the weights w, with $|| \cdot ||$ denoting the L_2 norm.

Another relevant feature implemented in XGBoost is data subsampling. In order to prevent overfitting, i.e., training too complex models that incorrectly model random noise as important parts of the models, only a random part of the n observations are used in the tree fitting process steps. As a convenient consequence, the computations are also speeded up. More details on XGBoost can be found in Chen T. Chen and Guestrin 2016.

The general framework of XGBoost works for any kind of response variable, provided that a suitable, twice-differentiable loss function is implemented. In the first part of our system, that deals with a binary classification problem (slippery / non-slippery), we will use a logistic loss function,

$$L(y_i, \hat{y_i}) = -y_i \log(\hat{y_i}) - (1 - y_i)\log(1 - \hat{y_i}) \tag{II.12}$$

where y_i is the true class for the observation i and $\hat{y_i}$ is the predicted probability of instance i to be of class 1, which is calculated as

$$\hat{y_i} = \frac{1}{1 + e^{-\hat{f}(\boldsymbol{x}_i)}}. \tag{II.13}$$

The logistic loss function (also called negative binomial log-likelihood and cross entropy loss) is the most common loss function for binary classification problems and is specifically convenient since it provides probabilities of a class instead of only the binary prediction. This is very useful when the consequences of misclassification is not the same for the two classes, which we will show further in Section II.4.1. The logistic loss function is also more robust to outliers than the exponential loss function, which the flight data might have due to error in sensor measurements.

In the second part of our system, we will have a continuous regression on the friction coefficient, so we use squared error as the loss function, namely

$$L(y_i, \hat{y_i}) = (y_i - \hat{y_i})^2, \tag{II.14}$$

where y_i is the true friction coefficient for instance i and $\hat{y_i} = \hat{f}(\boldsymbol{x}_i)$ is the predicted friction coefficient. For a continuous response the squared error loss is the most common and convenient loss function (Hastie, Tibshirani, and Friedmanl 2017), and therefore used here. Note that alternatives such as the Huber loss, that could be advantageous in terms of robustness against outliers, or the median loss are not usable since they are not twice-differentiable

The excellent performance of XGBoost, its scalability, and fast calculations are among the reasons why XGBoost was chosen to train the surface condition predictor in this work. In addition, as XGBoost is an ensemble of decision trees, its performance is not affected by multicollinearity (highly correlated explanatory variables) (Piramuthu 2008), which is highly present in our data. Especially between the variables created in Section II.2.3, which are different time variants of the same variable or a function of other existing variables, such as $\Delta_k x_i$. Another positive feature of XGBoost is that it handles missing data very well, because of the sparsity-aware split finding algorithm which creates default directions for the splits in the trees. This ensures that the models will continue to work in future scenarios where some measurements might be missing, which can be the case when working with sensor data. All these characteristics make XGBoost preferable to deep learning alternatives such as recurrent neural networks or LSTM networks. In addition, due to its decision trees base-learners, XGBoost is in general better at handling data of "mixed" types, better at handling missing values, are more robust to outliers, have a higher ability to deal with irrelevant variables, and are more interpretable than neural networks (Hastie, Tibshirani, and Friedmanl 2017). All of these are important factors when working with flight, snowtam and weather data.

II.3.2 Parameter tuning and model evaluation

When working with machine learning methods, parameter tuning is an important part of training the models. For example, finding good values for the penalties λ and γ are important to both prevent overfitting, which happens when λ and γ are too small, and underfitting, which happens when λ and γ are too large. Underfitting means training of too simple models that do not capture the data structures.

Model fitting, parameter tuning and model evaluation must be computed on different data. In this paper, we use a ten-fold nested cross validation, which is a method for model training, tuning and evaluation that is shown to provide an approximately unbiased estimate of the true model error (Varma and Simon 2006). The data are divided into ten folds, that are used in turn as a test set to evaluate the model trained in the other nine folds. The mean of the evaluation measure obtained in the ten test folds is regarded to be the performance of the model.

In each of the ten repetitions, the collected data from the nine training folds are again divided into tree folds to pursue a cross validated randomized search for tuning the parameters of the XGBoost model. The model is trained on two parts of the data with different combinations of parameters and evaluated on the third, which is repeated for all three folds. The parameters that give the best mean performance over all three folds are chosen. Five parameters are tuned with four settings for each of the parameters, where the settings are sampled from a distribution of possible values shown in Table II.4. This means that a total of 20 random combination of parameters are evaluated. A uniform distribution was selected for all parameter samplings, since we have no prior knowledge of the true best parameters. *n_estimators* is the number of decision trees in the model, that we indicated with K in the equations in Section II.3.1. *reg_lambda* and *min_split_loss* are the regularization parameters λ and γ respectively, *subsample* is the ratio of the data that is used in the data subsampling mentioned earlier, and *learning_rate* is ν, the step size shrinkage used at each boosting step.

II.3.3 Shapley Additive Explanations

The models created by XGBoost gets to be quite complex, as they combine scores from between 50 and 250 decision trees, making it difficult to understand how they makes their predictions. The increased use of black-box algorithms such as XGBoost and deep neural networks has escalated the focus on creating Explainable Artificial Intelligence (XAI) (Adadi and Berrada 2018). This involves methods for creating simpler explanation models, which are interpretable approximations of the complex black box models. There are a lot of reasons why it is important to have some understanding of how a system works. This includes gaining trust in the system, giving insight into how the system could be

Table II.4: Model parameters that where tuned together with the distributions they were sampled from.

Parameter	Explanation	Distribution
n_estimators	Number of trees	$\{50, 250\}$
reg_lambda	λ	$U(0, 10)$
min_split_loss	γ	$U(0, 0.4)$
subsample	Subsample ratio	$U(0.3, 1)$
learning_rate	Step size shrinkage	$U(0.1, 0.21)$

improved, allowing us to learn from the system, and monitoring possible errors in the data or models.

One method to get some insight into the decision basis of a machine learning system is by using SHAP (SHapley Additive Explanations), the state-of-the-art method for creating local explanations for machine learning models (S. M. Lundberg and S.-I. Lee 2017). Local explanations mean explaining why a specific observation got its prediction, which SHAP does by using Shapley Values from cooperative game theory (Shapley 2016). The variables are the players in the game, while the game is to predict if the runway conditions are slippery, or how slippery, in the case of the regression model. The goal of using shapley values is to distribute the prediction among the variables. This makes Shapley values part of the *additive feature attribution methods*, which means they have an explanation model that is a linear function of binary variables:

$$g(\mathbf{z}) = \phi_0 + \sum_{j=1}^{M} \phi_j z_j, \tag{II.15}$$

where $\boldsymbol{z} \in \{0,1\}^M$ is a coalition vector giving the absence / presence of the input variables in $\boldsymbol{x}$ and M is the number of variables in the original model. Methods with this explanation model assign an importance effect ϕ_j to each variable and summing the effects of all variables approximates the output of the original model. Several of the popular local explanation methods share this additive feature attribution method, such as *LIME* (Ribeiro, Singh, and Guestrin 2016), *DeepLIFT* (Shrikumar, Greenside, and Kundaje 2017), and *Layer-Wise Relevance Propagation* (Bach et al. 2015). The way Shapley values are calculated for variable j for a model $f(\boldsymbol{x})$ on observation i is:

$$\phi_j^{(i)} = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S\cup\{j\}}(\boldsymbol{x}^{(i)}_{S\cup\{j\}}) - f_S(\boldsymbol{x}^{(i)}_S)], \tag{II.16}$$

where $F \in \mathbb{R}^m$ is the set of all explanatory variables in the model and $\boldsymbol{x}_S$ is the values of the input features in the set S. Calculating the Shapley values requires training the model on all variable subsets $S \subseteq F$, and Eq. (II.16) sums up the marginal contribution of variable j by looking at all possible subsets without the variable and the effect of including it in these subsets.

To solve Eq. (II.16), S. M. Lundberg and S.-I. Lee 2017 proposed SHAP values, which are the shapley values of a conditional expectation function of the original model. In other words; SHAP values are the solution to Eq. (II.16) where $f_S(\boldsymbol{x}_S) = E[f(\boldsymbol{x})|\boldsymbol{x}_S]$ and S is the set of non-zero indexes in $\boldsymbol{z}$. This approximation of $f_S(\boldsymbol{x}_S)$ is done to account for the missing values in $\boldsymbol{x}_S$. SHAP values are theoretically optimal and are, according to Lundeberg & Lee,, the only possible consistent feature attribution method. But as a lot of theoretical optimums, they can be difficult to calculate. That is why S. Lundberg, C. Erion G., and al 2020; S. M. Lundberg, G. G. Erion, and S.-I. Lee 2018 derived an algorithm specific for tree ensembles that reduces the complexity of computing exact SHAP values for these kind of model structures. The algorithm is called *Tree SHAP* and is the explanation method used to explain the XGBoost models in this work.

As we are interested in knowing how our models work, we use the interventional approach to handle correlated variables. This means that we intervene on variables to break dependencies between dependent variables according to the rules of causal inference (Janzing, Minorics, and Bloebaum 2020). In practice, this is done by approximating $f_S(\boldsymbol{x}_S)$ with $E[f(\boldsymbol{x})|do(\boldsymbol{x}_S)]$ instead of $E[f(\boldsymbol{x})|\boldsymbol{x}_S]$, where do is Pearl 2000's do-operator. This operator simulates physical interventions by replacing certain functions or values from a model with a constant $X = x$, while keeping the rest of the model unchanged. The effect of this is that our explanations become true to the model instead of true to the data, which is further discussed in H. Chen et al. 2020.

II.4 Results and discussion

II.4.1 Performance of the classification model

As the dataset is highly imbalanced, with only 2.57% slippery landings, using accuracy as a performance evaluation metric for the binary classification is not a good option. Instead, the XGBoost classification model is evaluated by using confusion matrices, which show the amount of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) predictions, where slippery is regarded as positive and non-slippery as negative. The first two columns in Table II.5 show the confusion matrix for the predictions from the XGBoost classification model, where the columns are the predicted classes and the rows are the actual classes. As seen in Eq. (II.13), the output from the classification model are probabilities and not binary classifications, so the predictions are converted to binary classifications by using a threshold value for the probabilities. To account for the unbalanced dataset, 0.0257 was used as the threshold value, which is the expected value for the probabilities (this corresponds to the percentage of slippery conditions). However, the threshold value can be altered to account for the severity of making the two types of error, as will be seen later in this section.

Table II.5: Confusion matrices for the prediction from the different methods, where the highest number of TP and TN is marked in green and the lowest in red. S is the number of Slippery incidents, and NS in the number of Non-Slippery incidents.

		XGBoost		Runway		Scenario		Snowtam	
		S	NS	S	NS	S	NS	S	NS
Actual	S	**4 740**	423	**3 905**	1 258	4 223	940	4 006	1 157
	NS	28 863	166 482	46 967	148 378	78 894	**116 451**	20 679	**174 666**

To evaluate the performance of the XGBoost model, it is compared to the prediction from the runway model and the scenario model explained in Section II.1, as well as the reported surface conditions assessment in the Snowtam reports done by runway inspectors. The runway model is mainly implemented according the the paper by Klein-Paste, Bugge, and Huseby 2015, but includes the latest updates according to the operational IRIS system. One additional change has been carried out, which is removal of the rule that contamination coverage less than 10% automatically provides a braking action of 5, as we did not have a stable data source on this variable. As mentioned earlier, both the runway model and the Snowtam reports provide the surface conditions on a scale from 1-5. In Table II.5 we regard these methods to report slippery if the braking action is in the interval 1-3, meaning medium or less. The scenario model is implemented according to the paper by Huseby and Rabbe 2012 and is set to report slippery if it gives any warnings of slippery scenarios.

One observation from Table II.5 is that the models have different strengths and weaknesses. Since the scenario model is created to be a warning system, it has a high focus on identifying most of the slippery landings, even though some false warnings might happen. As a result, the scenario model gives a high amount of true slippery incidents, but misses as much as 40% of the non-slippery incidents. The runway model is more conservative than the scenario model. It gives a higher amount of true non-slippery incidents, but it misses the most on the slippery incidents. This contra-dictionary behavior could come from the motivation of the models, as they were initially created to be two parts of the same runway assessment system that fulfil each other. The assessment from the runway inspectors is the most conservative prediction, and is the method that gives the highest amount of true non-slippery landings. One reason why these assessments give more conservative predictions, could be the rarity in their updates. Good conditions are often more stable and can last for longer times, while difficult conditions can come and go more rapidly. The XGBoost model is optimized with the intention to balance the amount of true slippery and true non-slippery landings in the optimal way, and is the methods that gives the highest amount of true slippery landings, while at the same time gives a high amount of true non-slippery landings.

To see the difference in performance between the four methods more clearly, we use some commonly used performance evaluation metrics for imbalanced

Table II.6: Results from the prediction of slippery conditions from the different classification methods, where the highest and lowest value in every row is marked in green and red.

Metric	XGBoost	Runway	Scenario	Snowtam
Sensitivity	**0.918**	**0.756**	0.818	0.776
Specificity	0.852	0.760	**0.596**	**0.894**
G-Mean	**0.885**	0.758	**0.698**	0.833

datasets based on the confusion matrices:

- Sensitivity: Sensitivity $\frac{TP}{TP+FN}$ is the ratio of true positive predictions to the total amount of actual positive incidents, and gives the percentage of slippery incidents that were classified correctly.
- Specificity: Specificity $\frac{TN}{TN+FP}$ is the ratio of true negative predictions to the total amount of actual negative incidents, and gives the percentage of non-slippery incidents that were classified correctly.
- G-Mean: The geometric mean $\sqrt{Sensitivity * Specificity}$ is a combined metric that balances the sensitivity and the specificity.

The results of the prediction models and reported runway assessment in terms of these performance evaluation metrics are given in Table II.6. We see that XGBoost outperforms all the other methods in the amount of correctly classified slippery incidents with 92% sensitivity, while the runway model has the lowest sensitivity at 76%. XGBoost also outperforms the other two prediction models on correctly classifying the non-slippery landings, namely 85% of these, compared to 60% for the scenario model. But the conservative assessments from runway inspectors correctly classifies more of the non-slippery landings, namely 89%. The overall G-mean is still better for the XGBoost model, which has an improvement of 18% in true slippery incidents with only a 5% loss in true non-slippery incidents compared to the runway inspectors.

There is always a tradeoff between False Negatives (Type 1 error) and False Positives (Type 2 error), and the consequences of doing the two types of errors might be very different. Therefore, the severity of the consequences should be taken into account when evaluating the models' performances. As the models developed in this setting are primarily meant to work as warning systems, giving warnings when there might be slippery conditions, there is no doubt that avoiding Type 1 errors is the most important factor. If a pilot is not warned about actual bad runway conditions (Type 1 errors), accidents may happen. On the other hand, a warning system that gives too many warnings (Type 2 errors) might not be taken seriously. One benefit of the XGBoost model is that it predicts

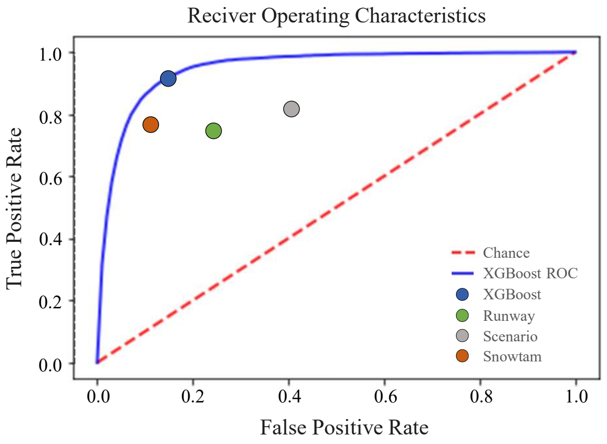


Figure II.1: Receiver Operating Characteristics curve for the XGBoost model together with the predictions from the different classification methods according to Table II.6. The closer the points/line is to the upper left corner, the better the performance.

the probability of a landing to be slippery. Using these probabilities, the user can decide the threshold value for landings to be regarded as slippery, thus altering the probability of the system to make the two different types of errors. A visualization of this is the Receiver Operating Characteristics (ROC) Curve, which plots the sensitivity (also called the True Positive Rate, TPR) vs. 1 - specificity (also called the False Positive Rate, FPR) for different threshold values. A plot of the ROC curve for the XGBoost model is given in Figure II.1, which shows that allowing a higher FPR provides a higher TPR.

A metric for measuring model performance using ROC curves is calculating the area under the curve (AUC). The area of 1 gives a perfect prediction, while the area of 0.5 (the area under the red dotted line in Fig. II.1) is a model as bad as random guessing. The XGBoost model achieves an area of 0.948, providing a high performance close to 1. The standard deviation in ROC AUC for the ten folds was 0.005, meaning we have quite consistent results with a relatively small variance in performance between the folds.

As the runway model, the scenario model, and the Snowtam reports gives direct classifications and not probabilities, it is not possible to create ROC curves from these methods. They have a fixed TPR and FPR and their performances are plotted as points in Figure II.1. The performance of the XGBoost classification model with the threshold used in Table II.5 and II.6 is plotted as a blue point on the ROC curve. We notice that all methods perform much better than random guessing, as they are long above the red dotted line. The prediction from XGBoost has both a higher TRP and a lower FPR than both the scenario model and the runway model. The reported runway assessments are also below the blue curve, meaning that XGBoost outperforms the reports, one only has to choose a threshold value according to the desired effect. If we want the XGBoost prediction to have the same TPR as the Snowtam reports (a sensitivity of 0.78), the XGBoost prediction has a specificity of 0.94, which is higher than the specificity of 0.89 for the Snowtam reports. The results show that the XGBoost model has indeed found patterns and relationships not covered by the knowledge- and engineering-based scenario model and runway model and outperforms them on all the metrics. The model also shows its usefulness when it not only matches human assessment from the runway inspectors, but actually exceeds it.

II.4.2 Performance of the regression model

For the friction limited landings, the friction coefficient reflects the amount of tire-pavement friction that was available. This means that we do not only know if it was slippery or not, we also know how slippery it was, and can use the estimated friction coefficients as a response variable when training the XGBoost model. Predicting the friction coefficient is done using the loss function given in Eq. (II.14) on the friction limited landings.

The performance of the XGBoost regression model is given in Table III.4 in the form of Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Braking Action Error (BAE), which are defined as

$$\mathrm{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}, \quad \mathrm{MAE} = \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{n},$$

$$\mathrm{BAE} = \sum_{i=1}^{n} \frac{|\mathrm{BA}(\hat{y}_i) - \mathrm{BA}(y_i)|}{n},$$

where $\hat{y}_i$ is the predicted friction coefficient, y_i is the true friction coefficient and $\mathrm{BA}(y_i)$ converts the friction coefficients to braking action using Table II.1. The MAE reflects the mean deviation of the predicted friction coefficient from the true friction coefficient, which is 0.0254 for the XGBoost regression model. The BAE reflects the mean number of braking action category the model misses with. As the runway model and the reported runway assessments give the predicted runway surface conditions only in braking actions and not in friction coefficients, RMSE and MAE cannot be obtained for these models, and we compare the models using the BAE. The scenario model only provides a binary classification (slippery / non-slippery) and not the level of slipperiness and is therefore not relevant in this setting.

We observe that the XGBoost regression model at average misses with approximately the half of one braking action (0.54). This is lower than both the prediction from the runway model and reported runway assessment from the Snowtam reports, which at average misses with 0.84 and 0.71, respectively. The exact distribution of deviation from the true braking action is given in Figure II.2. The deviation $\mathrm{BA}(\hat{y}_i) - \mathrm{BA}(y_i)$ shows the number of categories the prediction deviated from the conditions experienced during the landing. A deviation of zero means the prediction was correct, while a positive deviation

Table II.7: Mean results from the prediction of the friction coefficient from the XGBoost regression model together with the mean error in braking action for the runway model and the Snowtam reports.

Metric	XGBoost	Runway	Snowtam
RMSE	0.0332	-	-
MAE	0.0254	-	-
BAE	0.5402	0.8354	0.7124

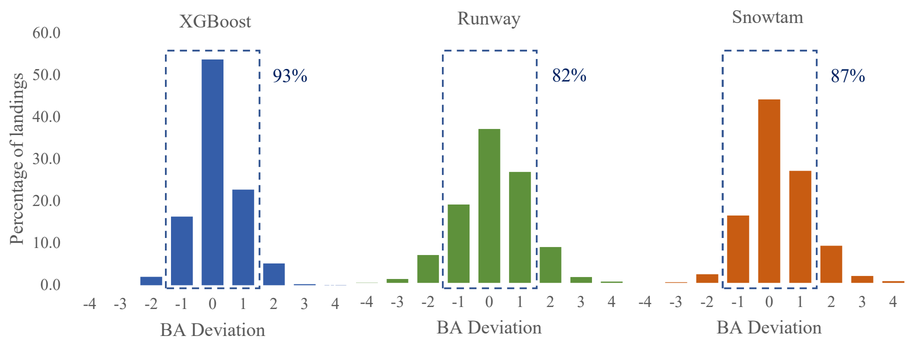


Figure II.2: The deviation of the predicted braking action from the estimated true braking action using XGBoost, the runway model and reported braking action from the Snowtam reports.

shows that the experienced conditions was worse than predicted. The deviations within ±1 is marked with blue dotted lines.

The figure shows that the XGBoost regression model has both a higher number of correctly classified landings (deviation 0) than the runway model and Snowtam reports, and has a higher percentage of the prediction within ±1 deviation. The regression model predicted 93% of the conditions within ±1, while the runway model and runway inspectors predicted this 82% and 87% of the times. XGBoost manages to outperform the other methods also when it comes to predicting the level of slipperiness.

II.4.3 Model discussions

The performance of the runway model in Figure II.2 corresponds quite closely with the performance given in the paper by Klein-Paste, Bugge, and Huseby 2015, where the runway model predicts 86% of the conditions within ±1 on a data set containing 1 261 friction limited landings in the winter seasons 2008/2009 to 2010/2011. This indicates that our alternations of the runway model described in Section II.4.1 did not have too much effect on the model performance. The performance of the assessment from runway inspections however, seems to have improved over the years, as they only had 77% of the conditions within ±1 in winter seasons 2008/2009 to 2010/2011 (Klein-Paste, Bugge, and Huseby 2015) compared to 87% over the seasons 2009/2010 to 2018/2019. The main reason for this is probably the increased focus at Norwegian airports to improve the quality of the runway assessment and runway reports over the last years, which seems to have been gainful.

There are several reasons why it is not possible to achieve a perfect AUC of 1.0 and BA Error of 0 for the XGBoost models, the most important being that both the explanatory variables $\boldsymbol{x}$ and the response variable y are subject to bias and measurement errors. As we are working with big data and several hundred thousand landings, it is not possible to investigate every flight, weather sensor measurement and Snowtam reports for errors. But we do know that measurement errors happen, especially in the sensors of the landing aircrafts, and the effect of this is discussed in detail in Midtfjord and Huseby 2020. In addition, difference in pilot behavior most probably have a contribution to inaccuracy in whether a landing is friction limited or slippery, as some pilots might brake harder and challenge the friction under the same circumstances as others might not. Another influential factor on the response variable could be the characteristics of the tires of the aircrafts, such as tire pressure, load, wear, and deformation (Niu et al. 2020). These are factors that affect the tire-to-pavement friction and could disturb the calculations of the friction coefficient from the flight sensors.

One factor that may have an effect on the runway surface conditions, is traffic volume and density. Including information about this in the explanatory variables could potentially increase the accuracy of the predictions. In the present study, however, the primary focus is on how the runway conditions are influenced by the weather. Thus, traffic volume and density data have not been included in the analysis.

It should be noted that the runway model and the assessment from the runway inspectors are created to be a 5 categories classification, and not a binary one, and their performances on the classification task should be seen in light of this. These models provide more information by giving the braking action for all landings, not only the friction limited ones. However, they do provide less information than the regression model for the friction limited landings, which gives a continuous prediction of the friction coefficient.

In order to handle the problem of the large amount of landings which are non-friction limited, it is possible to treat these as right censored data points, i.e., observations where only a lower limit of the friction coefficient is known instead of the precise value. Cases with censored data have been studied extensively in the literature, especially within survival analysis. Thus, there exists many well-established methods for statistical analysis of such data. By using these methods both friction limited and non-friction limited data can be part of the same regression. However, the high amount of censoring (96%) makes this a challenging task. In addition, there exists dependency between the friction coefficient and mechanisms controlling if a landing is friction limited, as both of these responses is explained by a lot of the same variables. Most standard survival analysis methods assume independence between the time-to-event distribution and the censoring distribution, and will give biased predictions on data involving dependent censoring. This problem will be addressed in a following paper.

A remark regarding the prediction of the braking action is a newly agreed transition in the international standardized Snowtam reports. The scales of braking action is currently transitioning from a five point scale to a six point scale (Kornstaedt 2021; Rodriguez 2019). Luckily, our models can easily be used with this new scale as well. Using the already trained XGBoost models, one only has to transform the predicted friction coefficients to the new braking action categories by using the new thresholds.

II.4.4 Global explanations of the models

SHAP values are created to give local explanations, meaning they provide information about why a single prediction happened. But with the high-speed estimations of SHAP values provided by Tree SHAP, it is possible to provide local explanations of entire datasets. Plotting local explanations for a whole test set provides information about how the model works as an entirety, for all the predictions. This means that the local explanations can be combined to give *global explanations* of the models. Figure II.3 is a plot of the local SHAP values across all test samples for the classification model, which combined creates a global explanation of how the classification model works for all predictions. To avoid showing ten plots (one for each test fold of the ten-fold cross validation), we showcase the SHAP values for the test fold of the ten-fold cross validation that gave prediction errors closest to the mean results of all ten repetitions. The figure is limited to the 20 variables with the highest sum of absolute SHAP values across the test set; $I_j = \sum_{i=1}^{n} |\phi_j^{(i)}|$, which is an indication of the importance of that variable to the model. The variables are displayed decreasing in importance from the top. An increase in SHAP value (towards the right on the x-axis) contributes to a higher probability of slippery conditions, and negative SHAP values contribute to lowering the probability. Note that when the scatter points do not fit on a line, they pile up to show density, and the color of each point represents the variable value of that individual point. SHAP values are given as a deviation from the expected value of the response $E[f(\boldsymbol{x})]$, which would be predicted if we do not condition on any variables. This means that a SHAP value of 0 indicates that including that variable would not influence the prediction at all. In Figure II.3, the SHAP values are given as deviation in every instance's scores obtained from all the trees of the model, which is the value given before taking the logistic function in Eq. (II.13).

One first observation from Figure II.3 is that *depth of contamination* is important for our model, and that the deeper the contamination, the higher the probability of slippery conditions. This corresponds to the fact that a higher amount of *accumulated dry snow* also contributes to more slippery conditions. Other factors that increase the probability of slipperiness is cold *runway temperature*, high *relative humidity*, and high *precipitation intensity*. These are all known factors that cause difficult landing conditions. One less intuitive result is that the presence of *damp* and *wet* contaminated runways make it less slippery. These most probably becomes surrogate variables explaining

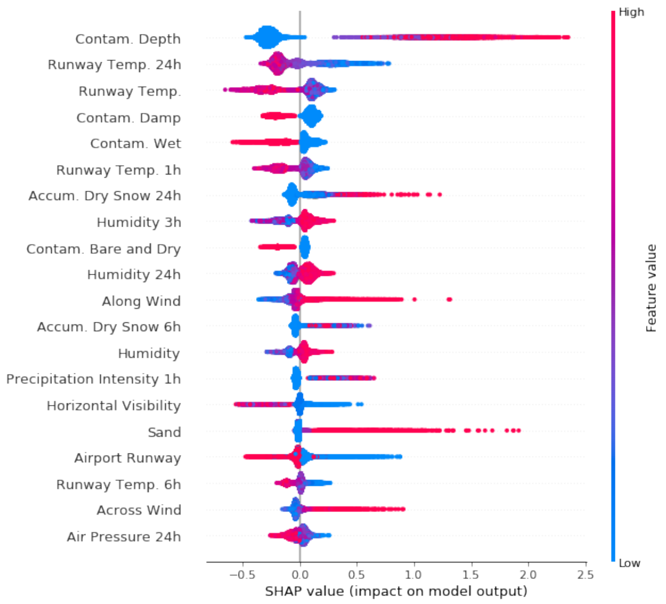


Figure II.3: Plot of the SHAP values across the test data for the classification model. Higher SHAP values correspond to an increase in the probability for the conditions to be slippery.

that there is no snow or ice on the runway, which often create more slippery conditions than just wet and damp runway. We also see that there is a difference between the two *airport runways* at Oslo Airport, that one seems to be more slippery than the other. When regarding the time of the observations, several variables with time difference up to 24 hours are important, as well as 1, 3 and 6 hours. It seems that the long-term effect of these variables affects the runway conditions, and that it is necessary to include such a wide timespan.

One interesting effect is that the presence of *sand* makes the model increase the probability of a landing to be slippery, even though the intention of sanding is the opposite. Since the runway operators only add sand to the runways in the presence of slippery conditions, XGBoost might use the presence of sand as a surrogate variable explaining slippery conditions caused by ice or snow on the runways. In addition, even though sanding can increase tire-pavement friction, especially when applied on solid contamination, it can also make it difficult to achieve the high levels of friction (Klein-Paste, Bugge, and Huseby 2015). The results from the XGBoost models and the SHAP values are entering the discussions around sanding of airport runways and might indicate that sanding is not always helpful in lowering the slipperiness.

Another observation is that *horizontal visibility* is an important variable. It is not intuitive why this should be an important variable for the experienced slipperiness for landing aircrafts, even though it of course affects the visual perception for the pilots. Sometimes it also indicates heavy precipitation. As Oslo Airport is located at a place where it is quite often foggy, a low horizontal visibility can be an indication of fog. Fog combined with cold weather conditions might lead to very slippery and dangerous runway conditions. This is also the reason why fog is involved in as much as two of Huseby and Rabbe 2012's eight slippery scenarios, namely *Freezing fog* and *Stratus/fog, air temperature below 0°C*. The former scenario happens when the temperature on the ground level drop to or below freezing point and the water droplets making up fog freeze on contact. This can result in black ice, which makes the runway very slippery.

As observed in the SHAP values for the classification model, the strength of *along wind* and *across wind* are part of the influential variables. These variables do not directly affect the available friction between the tires and the runway, but it does affect the necessary braking force for the landing aircraft. Along wind contributes with either a stopping force or pushing force dependent on the direction, and either increases or decreases the necessary force from the brakes of the aircraft. This could affect whether a landing is friction limited or not, as the pilot might have to brake harder. *Across wind* also contributes to difficulty in maneuvering and using more of the available friction on steering instead of braking. Therefore, you could say that the classification model works more like a landing condition predictor than a runway surface predictor, since it includes the overall experienced landing conditions for the aircrafts. This means that the model could also work for other kind of challenging landing conditions than snow and ice, e.g. wind and rain, and thereby be expanded to not-northern airports. However, even though the friction coefficient is universal and flight type indifferent, the effect that wind has on determining whether a landing is friction limited could vary with airplane size, weight and shape.

Figure II.4 shows the global SHAP values for the XGBoost regression model. The SHAP valus are given as deviation in the friction coefficient, and higher SHAP values corresponds to a higher friction coefficient, meaning less slippery conditions. The figure shows that the XGBoost regression model mostly uses the same variables as the classification model, but that the sequence has changed. For this model, the *accumulated dry snow* the last 24 hour is the most important factor, with *contamination depth* as the second. Some variables have increased their contribution significantly compared to the classification model. The predicted friction coefficient lowers with a low *horizontal visibility*, high *dew point*, and an increase in *air pressure*. We also see that the difference between the two *airport runways* are larger for the regression model than for the classification model.

One observation from the SHAP values for the regression model is that the effect of the along wind is opposite than for the classification model. Here stronger positive along wind contributes to an increase in the friction coefficient, even though it should not directly affect this, as the friction coefficient is a ratio of the frictional force between the tires and the runway. This counter-intuitive behaviour most probably comes from the calculation of the friction coefficient in the performance models mentioned in Section II.2.2, where along wind has a relationship with several of the variables that affect the estimation of the friction coefficient. As these relationships and effects are complex, we are not

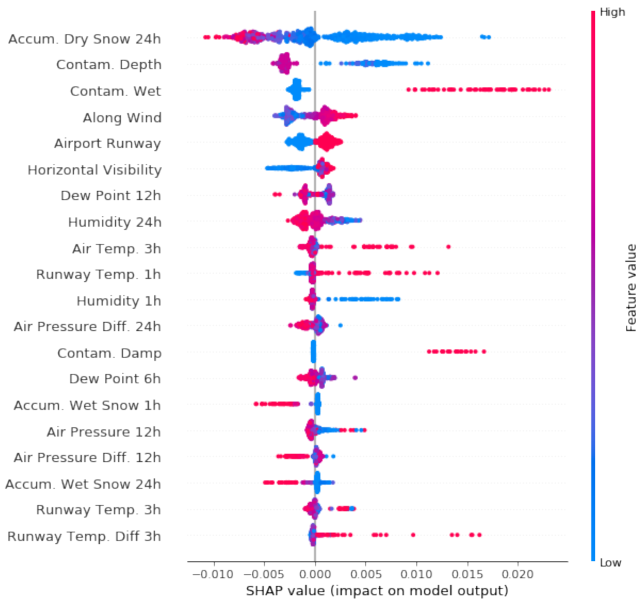


Figure II.4: Plot of the SHAP values across the test data for the regression model. Higher SHAP values correspond to an increase in the friction coefficient.

going to discuss it further in this paper. But XGBoost seems to pick up on these relationships and uses them, even though it might seem illogical when just looking at along wind isolated.

II.4.5 Creating autonomous models without Snowtam reports

We have shown that the XGBoost models manage to predict the experienced runway surface conditions better than the Snowtam reports created by the runway inspectors, both when classifying slippery / non-slippery conditions and when categorizing how slippery it is for the friction limited landings. Another point of interest is to check how good the models could work on their own without any human influence, to be an entirely autonomous system. This means only using data from the sensor variables (the meteorological data / weather data), and not the human assessments in the Snowtam reports.

Table II.8 shows a comparison of the XGBoost classification model and regression model including and excluding variables from the Snowtam reports, where X_{tot} is the total dataset of meteorological and Snowtam data and X_{met} is the subset of only meteorological data. For both the classification and regression model, there is a small decrease in performance when excluding data from the Snowtam reports. However, as the difference in performance is relatively small, we see that the models work quite well without information from the Snowtam reports. Even though we lose information about runway contamination and maintenance procedures, the XGBoost models seem to find other ways of

Table II.8: Comparison of the results from the XGBoost models using the total variable matrix and using only the meteorological variables, both in the classification (Clas.) and regression (Reg.) case.

		X_{tot}	X_{met}
Clas.	Sensitivity	0.918	0.916
	Specificity	0.852	0.850
	G-Mean	0.885	0.883
	ROC AUC	0.948	0.946
Reg.	RMSE	0.0332	0.0335
	MAE	0.0254	0.0257
	BA Error	0.5402	0.5448
	Error ±1	92,6%	92,3%

describing most of this information. This was substantiated by looking at the global SHAP plots for the models trained without the Snowtam reports, where especially accumulated dry snow, wet snow, and rain had significantly increased their contribution to the models, as well as runway temperature. The models were earlier dominated by contamination depth and type, but now the models use accumulated precipitation and runway temperature to explain the probable type and depth of contamination on the runways.

II.5 Local explanations and the decision support system

In high-risk applications such as air transportation, taking well informed and safe decisions is of main importance. Combining the XGBoost prediction models with SHAP local explanations can create a solid framework for a decision support system for runway conditions, to contribute to safer airplane landings and take-off. In section II.4.4, we plotted all local SHAP values together to create global explanations of the models. This is a strong tool to get some understanding of how the models work. However, just looking at the single local explanation for a prediction can be just as useful. For an user of an artificial intelligence system, only getting the final prediction might not be as helpful in itself, as it is difficult to trust a decision without any arguments. SHAP values give local explanations of every prediction, meaning we can get information about why the surface conditions were predicted as they were at all times.

Figure II.5 shows an example of local SHAP values for a prediction from the XGBoost regression model of the runway friction coefficient. The measurements that gave this prediction happened at the west runway at Oslo Airports at 8th February 2018, 22:23. The predicted friction coefficient of 0.1198 is lower than the expectation value. The main reason why XGBoost predicts this level of slipperiness is the presence of dry snow on ice with a depth of 8 mm, which is

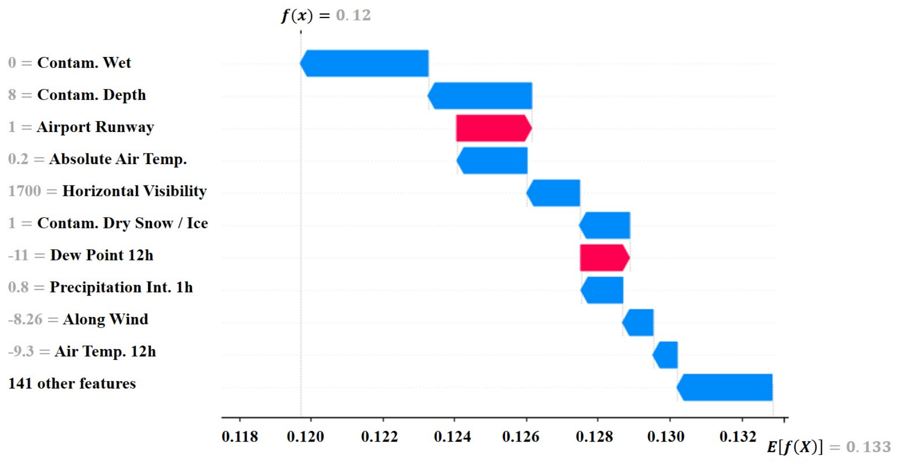
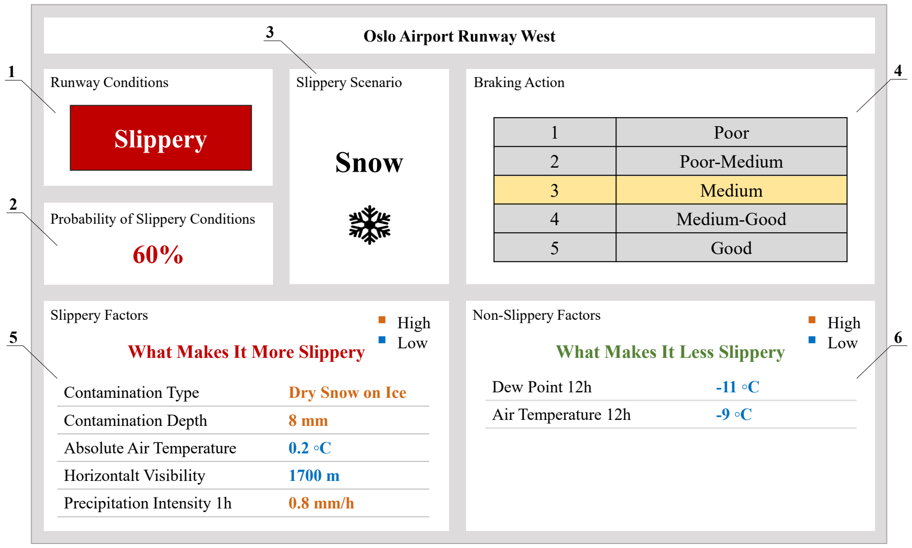


Figure II.5: Example of local SHAP values for a prediction of runway surface friction at Oslo Airport, where only the ten variables with largest absolute SHAP value are displayed. The values of the blue variables lower the predicted friction coefficient, while the red increase it.

given both as the absence of wet runways and the presence of dry snow on ice. This splitting of importance happens because of the one-hot encoding described in Section II.2.3. The absolute air temperature is almost zero, which can create quite difficult surface conditions. The horizontal visibility is quite low, so either there is fog or heavy precipitation. We can also see that there was precipitation quite recently. One factor that causes a prediction of less slippery conditions is the chosen runway west (Airport Runway 1), which seems to in general provide better landing conditions than the east runway according to the SHAP values. Another factor is the low dew point, which is way below the air temperature, so at least potential fog or air moisture will not condense and freeze on the runway.

An illustration of a decision support system, created based on the output from the XGBoost prediction models and the SHAP local explanations, is shown in Figure II.6. The system is meant to be used by airport operators in the same way as the IRIS system, based on the runway and scenario models discussed in (Klein-Paste, Bugge, and Huseby 2015) and (Huseby and Rabbe 2012), is used at 16 Norwegian airports today. The airport operators can use it as decision support in the logistic of airplane landing and take-off, when planning runway maintenance procedures, and when providing the most relevant information to the pilots.

Module 1 and 2 are the predictions from the classification model, where it is slippery if the probability of slippery conditions is higher than 50%. The probabilities are scaled to transform the expectation value of 0.0247 to 50%, to match the threshold used in Table II.5. Module 4 is the output from the regression model, converted to braking action according to Table II.1. Module 5



Oslo Airport Runway West

1 Runway Conditions

Slippery

2 Probability of Slippery Conditions

60%

3 Slippery Scenario

Snow

❄

4 Braking Action

1	Poor
2	Poor-Medium
3	Medium
4	Medium-Good
5	Good

5 Slippery Factors

🟧 High
🟦 Low

What Makes It More Slippery

Contamination Type	**Dry Snow on Ice**
Contamination Depth	**8 mm**
Absolute Air Temperature	**0.2 ◦C**
Horizontalt Visibility	**1700 m**
Precipitation Intensity 1h	**0.8 mm/h**

6 Non-Slippery Factors

🟧 High
🟦 Low

What Makes It Less Slippery

Dew Point 12h	**-11 ◦C**
Air Temperature 12h	**-9 ◦C**

Figure II.6: An illustration of a decision support system for airport runway conditions using the output from our models. Module 1 and 2 are the output from the classification model, module 3 is the output from the scenario model, model 4 is the output from the regression model, and module 5 and 6 are the output from the local explanations.

and 6 are outputs from the local explanations, which shows arguments for the prediction. The ten variables with highest SHAP value magnitude are given as arguments, where only up to five positive and five negative arguments are shown. The system shows the explanation of the classification model if the probability of slippery conditions is below 50% (non-slippery conditions), and shows the output from the explanations of the regression model if it is above 50% (slippery conditions). This way we make sure that the explanations focus on output from the model that is most trained within the given range. Module 3 is an additional feature to provide even more information and is an implementation of (Huseby and Rabbe 2012)'s scenario model, to provide more transparency by giving information about any potential slippery scenarios.

The illustration gives the same example as Figure II.5. At this time XGBoost classify the runway conditions as *Slippery* with a 60% probability, and that the braking action is medium. The user can see that the main reason for this level of slipperiness is because of dry snow on ice with a depth of 8mm, and that the air temperature is almost zero and the horizontal visibility is low. The SHAP values are given as text, as this is easier to comprehend for the end users. The system should have two separate interfaces, one for each runway, where the variable *airport runway* will be used to divide the predictions into two.

Including local explanations of the predictions provides a much more useful decision support system than only the prediction on its own, especially in critical systems as risk management, where trust is a crucial issue (B. Kim, Park, and Suh 2020). When using the decision support system in Figure II.6, the airport operators not only trust the system's decision seeing its decision basis, they can also check that the sensors and models work properly, and they can see what maintenance procedures to carry out to make it less slippery. The pilots can also be given information about what makes the landing conditions difficult and take this into consideration when planning a landing strategy.

There are some considerations to take into account when working with explainable artificial intelligence. One important point is multicollinearity between explanatory variables, which is a highly discussed problem in most model interpreting methods, as many of them assume independence between the variables (Ghosh and Ghattas 2015; Menze, Kelm, and Masuch 2009; Yan and D. Zhang 2015; Aas, Jullum, and Løland 2021). Both XGBoost and interventional SHAP values are robust to multicollinearity and including a lot of related variables should not affect their performance (H. Chen et al. 2020; Kangas, Heikinheimo, and Hippi 2015; Piramuthu 2008). One thing to bear in mind is that the interventional approach to SHAP is faithful to the prediction model, giving explanations of how the model works, not how the explanatory variables are connected to the response. One strength of XGBoost compared to other tree ensemble methods such as Random Forest, is that XGBoost in a much higher manner splits only on the most important variable in a group of highly correlated variables, not alternating between them. With other words: XGBoost has a build-in feature selector, which removes the need of an external feature selection process. However, this also means that variables which are highly correlated with the most relevant variables might get a very low feature importance, even though they could be highly related to the response. Since interventional SHAP are faithful to the model, the SHAP values in Figure II.3, II.4, II.5 and II.6 must be considered to explain how the XGBoost models work, not how the explanatory variables are related to the response.

II.6 Conclusions and future work

This paper presents a machine learning framework for providing real-time decision support for the assessment of airport runway conditions. This decision support system addresses the real-world problem within the aviation industry of efficiency and safety during winter seasons, which follows from the expected increased demand of air transportation.

The developed decision support system uses XGBoost to predict airport runway conditions, where the prediction models consist of a classification model to predict the presence of slippery conditions and a regression model to predict the level of slipperiness. The models are trained using weather data and runway reports and predict the runway conditions represented by the friction coefficient estimated using sensor data from landing aircrafts. The performance of the XGBoost models is compared to the state-of-the-art runway model and scenario model, as well as reported runway assessments from airport inspectors.

The XGBoost models achieve a high performance and outperform all the previous methods. This shows the strong abilities of machine learning to find and use patterns to model complex, physical phenomena when domain knowledge is included through the extraction of explanatory variables. An increased accuracy in the prediction of runway assessment can aid airport operators and pilots in making more appropriate decisions, which can contribute to avoiding accidents and lead to safer airplane landings.

The prediction models are combined with SHAP approximations to create interpretable models which can provide even more useful information. Combining the SHAP values with the prediction models provides a high accuracy and trustworthy decision support system, which presents arguments for the predicted slipperiness of the runway instead of only the prediction. In addition to contributing to safer and more economic operations of airport runways, providing trustworthy information about runway conditions can also contribute to lower fuel usage and less use of chemicals. If the runway conditions are known to be good, the pilots can use less fuel on thrust reverse, and the operators can use less anti-icing and de-icing chemicals on the runway.

Future work will be to expand the prediction system into a forecasting system to predict the runway conditions some hours into the future, by using time series and weather forecasts. This could help the airport operators to plan and execute necessary runway maintenance procedures, and in the logistic of airplane landings and take-off. We are also working on a novel, general machine learning method for right censored data which handles dependent censoring. With such a framework, we can take advantage of the measurements of minimum available friction from the airplane landings which are non-friction limited. Another important note is that the developed models are only trained and tested on data from Oslo Airport, and the effectiveness of these models on other airports needs to be evaluated. However, we are in the process of testing the framework on some other airports, both the finished trained model, as well as using the framework to train new, airport-specific models. It is also of interest to merge these two methods by using transfer learning methodology.

Acknowledgement

The authors are grateful to Avinor for making the flight data and runway data available for this research project. We also want to thank Boeing for invaluable help regarding the use of flight data.

References

Adadi, A. and Berrada, M. (2018). “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)”. In: *IEEE Access* vol. 6, pp. 52138–52160.

Adland, R. et al. (2021). “The value of meteorological data in marine risk assessment”. In: *Reliability Engineering & System Safety* vol. 209, p. 107480.

Alonso, J. et al. (2014). “On-board wet road surface identification using tyre/road noise and Support Vector Machines”. In: *Applied Acoustics* vol. 76, pp. 407–415.

Anupam, K. et al. (2017). “Finite Element Framework for the Computation of Runway Friction of Aircraft Tires”. In: *Transportation Research Record* vol. 2641, no. 1, pp. 126–138.

Bach, S. et al. (July 2015). “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: *PLOS ONE* vol. 10, no. 7, pp. 1–46.

Cerezo, V. et al. (2016). “Modelling-based approach to relate ground friction measurements to aircraft braking performance”. In: *Journal of Aircraft.*

Chen, H. et al. (2020). “True to the Model or True to the Data?” In: *arXiv.*

Chen, T. and Guestrin, C. (2016). “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD ’16. San Francisco, California, USA: Association for Computing Machinery, pp. 785–794.

Coffel, E. and Horton, R. (2015). “Climate change and the impact of extreme temperatures on aviation”. In: *Weather, Climate, and Society* vol. 7, no. 1, pp. 94–102.

De Coste, M. et al. (2021). “A hybrid ensemble modelling framework for the prediction of breakup ice jams on Northern Canadian Rivers”. In: *Cold Regions Science and Technology* vol. 189, p. 103302.

Erdogan, G., Alexander, L., and Rajamani, R. (2011). “Estimation of Tire-Road Friction Coefficient Using a Novel Wireless Piezoelectric Tire Sensor”. In: *IEEE Sensors Journal* vol. 11, no. 2, pp. 267–279.

Foundation, F. S. (n.d.). *Aviation Safety Network.* URL: https://aviation-safety.net.

Friedman, J. H. (2001). “Greedy function approximation: A gradient boosting machine.” In: *The Annals of Statistics* vol. 29, no. 5, pp. 1189–1232.

Ghosh, J. and Ghattas, A. E. (2015). “Bayesian Variable Selection Under Collinearity”. In: *The American Statistician* vol. 69, no. 3, pp. 165–173.

Giesman, P. (2005). “Wet runways, physics, certification, application”. In: *Boeing Performance and Flight Operations Engineering Conference*, no. 8, pp. 1–24.

Gultepe, I. et al. (2019). “A review of high impact weather for aviation meteorology”. In: *Pure and applied geophysics* vol. 176, no. 5, pp. 1869–1921.

Gőssling, S. and Humpe, A. (2020). “The global scale, distribution and growth of aviation: Implications for climate change”. In: *Global Environmental Change* vol. 65, p. 102194.

Hastie, T., Tibshirani, R., and Friedmanl, J. (2017). *The Elements of Statistical Learning.* Springer.

He, J., Hao, Y., and Wang, X. (2021). “An Interpretable Aid Decision-Making Model for Flag State Control Ship Detention Based on SMOTE and XGBoost”. In: *Journal of Marine Science and Engineering* vol. 9, no. 2.

Holzmann, F. et al. (2006). “Predictive estimation of the road-tire friction coefficient”. In: *2006 IEEE Conference on Computer Aided Control System*

Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, pp. 885–890.

Huseby, A. B. and Rabbe, M. (2012). “A scenario based model for assessing runway conditions using weather data”. In: *11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference*. Ed. by PSAM and ESREL. Curran Associates, Inc, pp. 5092–5101.

— (2018). “Optimizing warnings for slippery runways based on weather data”. In: *Safety and Reliability - Safe Societies in a Changing World. Proc. ESREL 2018*.

Janzing, D., Minorics, L., and Bloebaum, P. (2020). “Feature relevance quantification in explainable AI: A causal problem”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Chiappa, S. and Calandra, R. Vol. 108. Proceedings of Machine Learning Research. PMLR, pp. 2907–2916.

Jiao, Z., Sun, D., et al. (2019). “A high efficiency aircraft anti-skid brake control with runway identification”. In: *Aerospace Science and Technology* vol. 91, pp. 82–95.

Jiao, Z., Wang, Z., et al. (2021). “A novel aircraft anti-skid brake control method based on runway maximum friction tracking algorithm”. In: *Aerospace Science and Technology* vol. 110, p. 106482.

Jin, M. et al. (2019). “Oil tanker risks on the marine environment: An empirical study and policy implications”. In: *Marine Policy* vol. 108, p. 103655.

Joshi, K. et al. (2015). “Braking Availability Tester for Realistic Assessment of Aircraft Landing Distance on Winter Runways”. In: *Journal of Aerospace Engineering* vol. 28, no. 4.

Juga, I., Nurmi, P., and Hippi, M. (Sept. 2013). “Statistical modelling of wintertime road surface friction”. In: *Meteorological Applications* vol. 20, pp. 318–329.

Kangas, M., Heikinheimo, M., and Hippi, M. (Jan. 2015). “RoadSurf: A modelling system for predicting road weather and road surface conditions”. In: *Meteorological Applications* vol. 22, pp. 544–553.

Kellner, L. et al. (2019). “Establishing a common database of ice experiments and using machine learning to understand and predict ice behavior”. In: *Cold Regions Science and Technology* vol. 162, pp. 56–73.

Khaleghian, M., Emami, A., and Taheri, S. (May 2017). “A technical survey on tire-road friction estimation”. In: *Friction* vol. 5.

Kim, B., Park, J., and Suh, J. (2020). “Transparency and accountability in AI decision support: Explaining and visualizing convolutional neural networks for text information”. In: *Decision Support Systems* vol. 134, p. 113302.

Kim, S., Lee, J., and Yoon, T. (2021). “Road surface conditions forecasting in rainy weather using artificial neural networks”. In: *Safety Science* vol. 140, p. 105302.

Klein-Paste, A., Huseby, A. B., et al. (2012). “Braking performance of commercial airplanes during operation on winter contaminated runways”. In: *Cold Regions Science and Technology*, no. 79–80, pp. 29–37.

Klein-Paste, A., Bugge, H. J., and Huseby, A. B. (June 2015). “A decision support model to assess the braking performance on snow and ice contaminated runways”. In: *Cold Regions Science and Technology* vol. 48.

Kongrattanaprasert, W. et al. (2009). “Detection of Road Surface Conditions Using Tire Noise from Vehicles”. In: *IEEJ Transactions on Industry Applications* vol. 129, no. 7, pp. 761–767.

Kornstaedt, L. (2021). “Global Reporting Formal”. In: *Webinar on the Global Reporting Format (GRF).*

Lee, C., Hedrick, K., and Yi, K. (2004). “Real-time slip-based estimation of maximum tire-road friction coefficient”. In: *IEEE/ASME Transactions on Mechatronics* vol. 9, no. 2, pp. 454–458.

Li, S. et al. (2020). “Fast Evaluation of Aircraft Icing Severity Using Machine Learning Based on XGBoost”. In: *Aerospace* vol. 7, no. 4.

Lundberg, S., Erion G., C., and al, H. et (2020). “From local explanations to global understanding with explainable AI for trees”. In: *Nature Machine Intelligence* vol. 2, pp. 56–67.

Lundberg, S. M., Erion, G. G., and Lee, S.-I. (2018). “Consistent individualized feature attribution for tree ensembles”. In: *arXiv preprint.*

Lundberg, S. M. and Lee, S.-I. (2017). “A Unified Approach to Interpreting Model Predictions”. In: *NIPS.*

Menze, B., Kelm, B., and Masuch, R. e. a. (2009). “A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data”. In: *BMC Bioinformatics* vol. 10, no. 213.

Midtfjord, A. and Huseby, A. B. (2020). “Estimating Runway Friction Using Flight Data”. In: *e-proceedings of the 30th European Safety and Reliability Conference and 15th Probabilistic Safety Assessment and Management Conference (ESREL2020 PSAM15).* Ed. by Baraldi, P., Maio, F. P. D., and Zio, E. Research Publishing Services.

Montewka, J. et al. (2015). “Towards probabilistic models for the prediction of a ship performance in dynamic ice”. In: *Cold Regions Science and Technology* vol. 112, pp. 14–28.

Niskanen, A. J. and Tuononen, A. J. (2014). “Three 3-axis accelerometers fixed inside the tyre for studying contact patch deformations in wet conditions”. In: *Vehicle System Dynamics* vol. 52, no. sup1, pp. 287–298.

Niu, Y. et al. (July 2020). “Estimation for Runway Friction Coefficient Based on Multi-Sensor Information Fusion and Model Correlation”. In: *Sensors* vol. 20, p. 3886.

Parsa, A. B. et al. (2020). “Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis”. In: *Accident Analysis & Prevention* vol. 136, p. 105405.

Pearl, J. (2000). *Causality.* Cambridge University Press.

Piramuthu, S. (2008). “Input data for decision trees”. In: *Expert Systems with Applications* vol. 34, no. 2, pp. 1220–1226.

Putov, V. V., Putov, A. V., Sheludko, V. N., et al. (2015). “On correlation between the airport runway friction coefficient measurement results and the real-life aircraft take-off and landing braking Characteristics". In: *2015 XVIII International Conference on Soft Computing and Measurements (SCM)*, pp. 119–121.

Putov, V. V., Putov, A. V., Kazakov, V. P., et al. (2015). "On improving the efficiency of methods and technical solutions of prelanding air field coatings frictional properties control". In: *2015 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW)*, pp. 270–274.

Rado, Z. and Wambold, J. (2014). "Correlation of ground friction measurements to aircraft braking friction calculated from flight data recorders". In: *In Proceedings of the 4th International Safer Roads Conference*. Cheltenham, UK, pp. 18–21.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, pp. 1135–1144.

Rodriguez, A. (Aug. 2019). "Runway Assessment Matrix (RCAM)". In: *ICAO SAM Regional Seminar on the GRF for Runway Conditions*.

Shapley, L. S. (2016). "17. A Value for n-Person Games". In: *Contributions to the Theory of Games (AM-28), Volume II*. Ed. by Kuhn, H. W. and Tucker, A. W. Princeton University Press, pp. 307–318.

Shen, X. and Wei, S. (2020). "Application of XGBoost for Hazardous Material Road Transport Accident Severity Analysis". In: *IEEE Access* vol. 8, pp. 206806–206819.

Shi, X. et al. (2019). "A feature learning approach based on XGBoost for driving assessment and risk prediction". In: *Accident Analysis & Prevention* vol. 129, pp. 170–179.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). "Learning Important Features Through Propagating Activation Differences". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Precup, D. and Teh, Y. W. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 3145–3153.

Tuononen, A. (2008). "Optical position detection to measure tyre carcass deflections". In: *Vehicle System Dynamics* vol. 46, no. 6, pp. 471–481.

Varma, S. and Simon, R. (Feb. 2006). "Bias in Error Estimation When Using Cross-Validation for Model Selection." BMC Bioinformatics, 7(1), 91". In: *BMC bioinformatics* vol. 7, p. 91.

Yan, K. and Zhang, D. (2015). "Feature selection and analysis on correlated gas sensor data with recursive feature elimination". In: *Sensors and Actuators B: Chemical* vol. 212, pp. 353–363.

Zhang, W. et al. (Sept. 2021). "Fusion and analysis of data sources for assessing aircraft braking performance on non-dry runways". In.

Zhang, X. and Mahadevan, S. (Feb. 2017). "Aircraft re-routing optimization and performance assessment under uncertainty". In: *Decision Support Systems* vol. 96, pp. 67–82.

Zhu, Y. et al. (2021). “Review on flashover risk prediction method of iced insulator based on icing monitoring technology”. In: *Cold Regions Science and Technology* vol. 185, p. 103252.

Aas, K., Jullum, M., and Løland, A. (2021). “Explaining individual predictions when features are dependent: More accurate approximations to Shapley values”. In: *Artificial Intelligence* vol. 298, p. 103502.

Appendix II.A List of variables

Table II.9: Full list of variables used in the prediction models. These add up to 151 variables after taking time lags, trends, accumulation and one-hot-encoding.

Snowtam	Meteorological data			
	Observation	Lag [1,3,6,12,24]	Trend [1,3,6,12,24]	Accum. [1,3,6,12,24]
Sand	Precip. Intensity	Precip. Intensity		
Warm Sand	Air Temp.	Air Temp.		
Deice	Runway Temp.	Runway Temp.	Runway Temp.	
Aice	Relative Humidity	Relative Humidity	Relative Humidity	
Contam. Depth	Air Pressure	Air Pressure	Air Pressure	
Contam. Coverage	Dew Point	Dew Point		
Contam. Type	Horizontal Visibility	Horizontal Visibility		
	Precip. Type	Precip. Type		
	Dry Snow			Dry Snow
	Wet Snow			Wet Snow
	Sleet			Sleet
	Rain			Rain
	Wind Direction			
	Max. Wind Speed			
	Mean Wind Speed			
	Along Wind Speed			
	Across Wind Speed			
	Abs. Air Temp.			
	Abs. Runway Temp.			
	Airport Runway			