JOHANNES KEPLER
UNIVERSITY LINZ

Author
Christoph Großauer, BSc

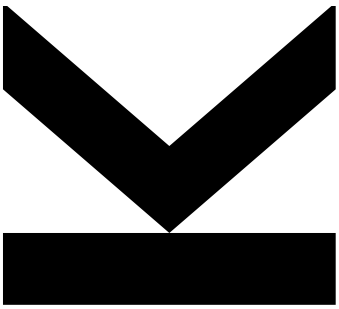
Submission
Institute of Business Informatics - Data & Knowledge Engineering

Thesis Supervisor
Assoz.-Prof. Dr. Christoph Schütz

Assistant Thesis Supervisor
Simon Staudinger, MSc

July 2026

Assessing the Reliability of Flight Delay Predictions Using the Perturbation Approach



Master Thesis

to confer the academic degree of

Master of Science

in the Master's Program

Business Informatics

JOHANNES KEPLER UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at

Preface

The results of this thesis contributed to the following publication in the Proceedings of the 27th International Conference on Enterprise Information Systems:

Staudinger, S., Großauer, C., Badzura, P., Schuetz, C. G. and Schrefl, M. (2025). Implementing the Perturbation Approach for Reliability Assessment: A Case Study in the Context of Flight Delay Prediction. In Proceedings of the 27th International Conference on Enterprise Information Systems - Volume 1: ICEIS; ISBN 978-989-758-749-8; ISSN 2184-4992, SciTePress, pages 75-86. DOI: 10.5220/0013299700003929.

Abstract

Flight delays are known to be one of the biggest disturbance factors in the air travel domain. Therefore, solving this logistical problem is critical for stakeholders in the aviation industry. With the rise of machine learning (ML) and artificial intelligence (AI), predicting flight delays has become more reliable than ever. Although the introduction of AI and ML allowed for a more precise flight delay prediction, additional issues arose due to factors like model complexity, high volumes of data, complex data gathering and processing as well as black box algorithms. Furthermore, predictions of ML systems are often challenging to reproduce without significant insight into the development process. Additionally, some ML algorithms in classification tasks only provide the end-user a target class without specifying possibilities to evaluate how certain the classification is. Therefore, edge cases are potentially not recognised as such, leading to suboptimal decisions with possible ramifications. Because of these issues, evaluating the reliability of individual predictions is critical to prevent potential damages of decisions based on unreliable predictions and increase trust in the predictive system. One approach to identify potentially unreliable predictions is the perturbation approach. The perturbation approach involves slightly altering input vectors within predefined margins to observe whether minor changes in input vectors lead to unintended changes to the prediction outcome. This approach is embedded into a reliability assessment process that follows the Cross-Industry Standard Process for Data Mining (CRISP-DM), extended with additional documentation tasks needed for reliability assessment. This reliability assessment process combined with the perturbation approach allows stakeholders to evaluate the certainty of individual predictions without requiring in-depth knowledge of the underlying machine learning algorithm, creating a model-agnostic method for assessing prediction outcomes. This thesis describes a data mining process for multi-class prediction of flight arrival delays to test the validity of the reliability assessment process on a real-world dataset. The data mining project follows the reliability assessment process to find sensitive data fields on basis of which perturbation is used to find unreliable predictions. Three differing tree-based classification models have been trained. The evaluated Random Forest Classifier performed best with an accuracy of 75.19% over all classes. By using perturbation analysis to filter the input dataset to only contain entries leading to reliable predictions, accuracy changed to 80.07%, indicating that perturbation is capable of identifying unreliable predictions regarding flight arrival delay classification. In the theoretical scenario that all unreliable predictions could be corrected with domain knowledge, the test accuracy of the model would amount to 84.10%, showing the potential positive influence of the perturbation approach on the predictive system.

Kurzfassung

Verzögerungen im Flugverkehr gelten als einer der größten Störfaktoren in der Luftfahrtbranche. Daher ist eine zeitgerechte Identifikation von Verzögerungen wesentlich für einen effizienten Ablauf aller Prozesse innerhalb der Branche. Mit dem Aufstieg von maschinellem Lernen (ML) und künstlicher Intelligenz (KI) bekamen Unternehmen weitere Möglichkeiten, Flugverspätungen zuverlässiger als je zuvor vorherzusagen. Jedoch führen Modelle auf Basis von ML und KI teils zu unerwarteten und nicht replizierbaren Vorhersagen, die das Vertrauen in Vorhersagemodelle verringern. Des Weiteren wird bei klassifizierenden Vorhersagemodellen ein Benutzer nicht über die Verlässlichkeit der Vorhersage informiert. Speziell in Branchen, die auf verlässliche Vorhersagen angewiesen sind, ist es besonders wichtig, die erstellten Vorhersagen auf deren Verlässlichkeit zu überprüfen. Eine solche Methode, die Verlässlichkeit einzelner Vorhersagen zu überprüfen, ist der 'Perturbation Approach'. Bei dem Einsatz dieser Methode werden Datenvektoren, die für eine Vorhersage an ein Vorhersagemodell gegeben werden, leicht gestört, um die Auswirkung der Veränderung auf die Vorhersage zu untersuchen. Sind solche Störungen genug, um die Vorhersage zu beeinflussen, gilt die Vorhersage als unzuverlässig. Diese Methode wurde in einen Prozess für die Erkennung von unzuverlässigen Vorhersagen (Relaibility Checking Process) eingebettet, welcher streng mit dem 'Cross-Industry Process for Data Mining' (CRISP-DM) verbunden ist. Durch dieses Vorgehen ist es möglich, die Verlässlichkeit von individuellen Vorhersagen auf eine projektspezifische Art zu bestimmen, die unabhängig von den verwendeten Vorhersagemodellen ist. Im Zuge der Arbeit wurden drei Modelle für die Klassifizierung von Flugverspätungen erstellt, die Flugdaten in verfrühte, pünktliche und verspätete Ankunft in KATL (Atlanta, USA) klassifizieren. Für die Entwicklung der Modelle wurde der 'Reliability Checking Process' angewandt und alle Schritte dokumentiert. Die Vorhersagen wurden mittels 'Perturbation' auf deren Verlässlichkeit überprüft, um Schlüsse über die Verwendbarkeit des 'Perturbation Approach' innerhalb der Luftfahrbranche ziehen zu können. Das beste Vorhersagemodell erreichte eine 'Accuracy' von 75.19% über alle Klassen. Durch die Erkennung unzuverlässiger Vorhersagen und die Eliminierung solcher würde das Modell eine 'Accuracy' von 80.07% erreichen. Des Weiteren erlaubt die Identifikation von unzuverlässigen Vorhersagen eine manuelle Analyse von Datenvektoren, die zu unzuverlässigen Vorhersagen führen. Durch domänenspezifische Kenntnisse könnte ein Benutzer die Vorhersage manuell ausbessern. Theoretisch könnte ein Benutzer alle als unzuverlässig markierten Einträge manuell auf die tatsächlich korrekte Vorhersage ausbessern, was eine 'Accuracy' von 84.10% im ausgebesserten Datensatz zur Folge hätte und somit die Wirksamkeit des 'Perturbation Approach' innerhalb dieser Arbeit aufzeigt.

Contents

List of Figures

List of Tables

1

Introduction

According to the Bureau of Transportation Statistics in the United States of America, the number of passengers within the aviation industry was steadily increasing from 700 million in the year 2002 to 1052 million in 2019 [1]. Although the COVID-19 pandemic reduced air travel in the United States by 96% in April 2020 [2], the number of civil flights quickly climbed back to pre-pandemic levels in 2023, increasing steadily ever since [3]. With an increase in traffic, it becomes more challenging for airline companies to manage their flight schedules. Therefore, the need to analyse and predict aviation traffic increases in urgency. Especially, correctly estimating and anticipating flight delays is one of the key functions for companies within the aviation industry for cost reduction. A study conducted by the Federal Aviation Administration (FAA) evaluated that in the year 2007 alone, arrival delays cost the seven major airlines in the USA $4.6 billion [4]. Additionally, the FAA estimated in their study that passengers had to spend around $16.7 billion more due to the direct or indirect effects of delayed or missed connecting flights [4]. Although the study itself acknowledges that passenger cost figures are likely to be generous estimates, the effects on the economy and trust in aviation are shown to be significant [4].

With the rise of machine learning (ML) and artificial intelligence (AI), researchers are increasingly relying on these technologies to improve the accuracy of flight delay estimations. These estimates should give operators higher confidence in the accuracy of flight schedules compared to estimates produced without the use of ML and AI. However, some ML solutions are not entirely reproducible due to (i) the algorithm's blackbox characteristics, (ii) the lack of source code provided, (iii) incomplete data gathering and preprocessing documentation or (iv) data leakage in the training and preprocessing phases of the ML project [5]. This reproducibility problem introduces the need for new methods to structure and evaluate machine learning models with respect to all phases of the model's development. Staudinger et al. [6] propose perturbation analysis to promote better documentation and identify problems with resulting model predictions. The process was developed to assess the reliability of individual predictions by observing impacts of slight deviations in input data on predictive outcomes. Evaluating reliability of individual predictions is an often overlooked aspect within the machine learning domain. As predictive systems grow in popularity, especially in domains with narrow error margins, accuracy of predictions and the trust in predictive outcomes gains more and more relevance [7]. Besides indicating reliability of individual predictions, the process proposed by Staudinger et al. [6] allows to better identify border cases of machine learning classification results for stakeholders not included in the development process of ML projects. In order to better understand the usability of the perturbation approach and its risks and benefits,

this thesis was conducted to evaluate the approach on a real-world problem.

This thesis applies the reliability assessment process of Staudinger et al. [6] to the aviation sector, investigating potential benefits of such an application. The study follows a design-science research approach, where the primary artifact consists of a structured process for developing machine learning models on which reliability assessment on flight delay predictions are conductible. The proposed process is evaluated based on impacts on the produced model's performance metrics, potential to detect unreliable predictions and implications on documentation and development tasks. Model development and validation follows the Cross-Industry Standard Process for Data Mining (CRISP-DM), extended with the reliability-specific alterations proposed by Staudinger et al. [6].

The structure of this thesis is as follows. Chapter 2 provides a comprehensive introduction to the essential topics needed to understand the model development and perturbation processes. Chapter 3 outlines the stages of the CRISP-DM framework, serving as the documentation for the model-building phases. Section 3.1 provides insight into domain specific circumstances from which assumptions for subsequent tasks are derived. Section 3.2 documents the source and structure of gathered data. Section 3.3 explains decisions regarding preprocessing steps to obtain a clean dataset. Section 3.4 introduces algorithms used for model training and the processes to find fitting model parameters. Finally, Section 3.5 declares target baselines for evaluation metrics and provides insight into the trained models' metrics. Furthermore, key findings relevant to the perturbation approach are highlighted during all sections of Chapter 3, which form the foundation for the subsequent chapters. Chapter 4 looks into the mechanics of the perturbation approach and presents the corresponding results. Chapter 5 discusses the implications of the perturbation approach, addressing its limitations and the constraints of the thesis as a whole. Additionally, it offers insights into potential future directions within this field.

2

Background

The reliability checking reference process as defined by Staudinger et al. [6] is a process, that closely follows the CRISP-DM reference model. To successfully conduct this analysis, additional observation and documentation steps were added to the CRISP-DM process across all stages of a given data mining project's development. This additional documentation forms the basis for conducting perturbation analysis during a models' deployment stage. Since the proposed process is a novel idea and necessary to understand the motivation behind this thesis, this section provides detailed insight into the process defined by Staudinger et al. Given that CRISP-DM is strongly intertwined with the reliability assessment process, this section also briefly explains the reference model, particularly in the context of the aviation domain, due to its relevance for this thesis. Lastly, Section 2.3 introduces the works on which this thesis is based on and provides an overview of current works within the field of flight delay prediction and reliability assessment.

2.1 CRISP-DM

The Cross-Industry Standard Process for Data Mining (CRISP-DM) was developed to improve the quality and structure of data mining processes, while also increasing stakeholder confidence in data mining projects [8]. Currently, data mining and machine learning projects are already widespread, however, the CRISP-DM process and similar reference models remain highly effective [9]. Although shortcomings like weak interdependency between process phases, lack of data privacy and regulation compliance in its traditional definition have been identified, CRISP-DM provides valuable guidelines for data mining and machine learning projects [10].

CRISP-DM is an iterative process consisting of six distinct phases. In each phase, key findings are documented, forming the foundation for further phases in CRISP-DM, which is shown in Figure 2.1. This data-centric process iterates until the defined and refined goals regarding the respective domain's needs are met and the developed artifact is embedded into the intended system. In the first phase, *Business Understanding*, key aspects of the overarching domain are identified based on which initial project goals are established. The *Data Understanding* phase involves collecting data and analysing statistical properties and distributions. In the *Data Preparation* phase, the collected dataset is cleaned, transformed and integrated to produce a dataset ready for modelling operations. The *Modelling* phase consists of the development and training processes for the intended artifact. In the *Evaluation* phase, the development team compares the target goals with the actual achieved goals. Finally, the *Deployment* phase involves embedding the developed artifact into production

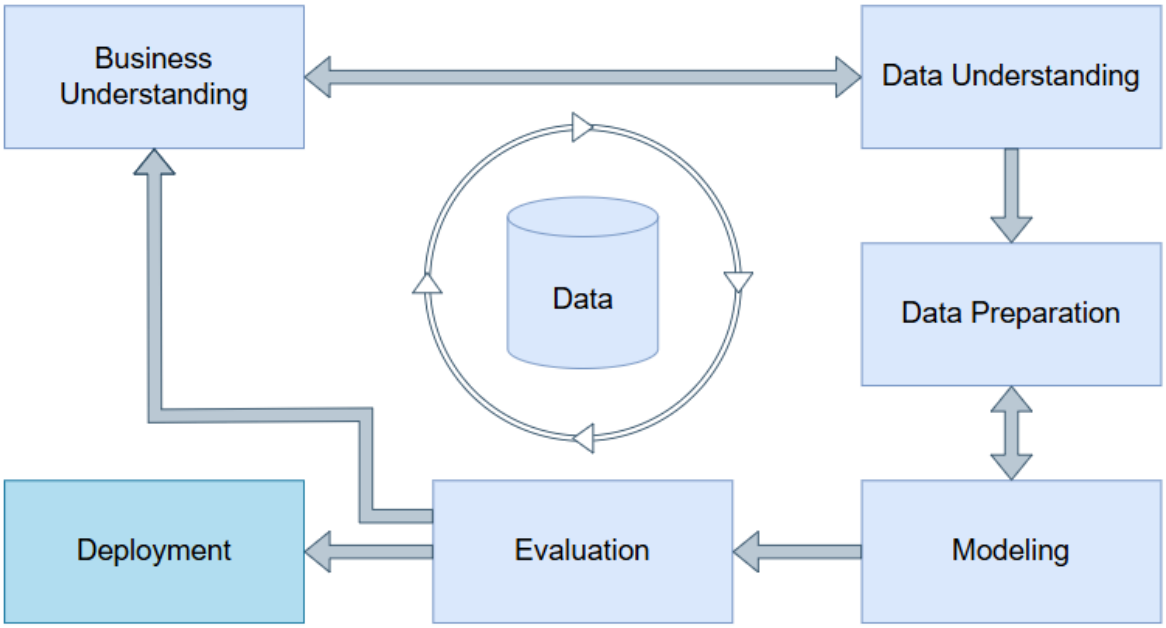


Figure 2.1: CRISP-DM phases and process sequence. Adapted from [8].

and evaluating the overall success of the process. At the centre of the process lies the data which influences each decision going forward. Additionally, all process phases must continuously take previous observations and domain specific problems into account to successfully reach declared project goals [8].

2.2 Reliability Assessment Reference Process

Machine learning and data mining projects often evaluate models using global metrics such as *accuracy*, *precision*, and *recall*. However, the reliability of individual predictive outcomes cannot be guaranteed by high accuracy, for example. Given a predictive scenario, where slight alterations of the model input are enough to change the prediction outcome, the scenario can be considered unreliable. Furthermore, binning and regression functions may introduce vagueness into related features, while sensor data is subject to physical limitations of hardware and associated sensor tolerances. In such cases, the true value of a given feature might be unknown within narrow margins. If an input feature changes within these margins, a reliable predictive scenario should not change its outcome. To identify such unreliable predictions in classification problems, Staudinger et al. [6] extended the CRISP-DM process by adding documentation requirements and tasks as seen in Figure 2.2 to allow for structured processes and explainable definitions for perturbation options in the familiar environment that is CRISP-DM for many data mining and machine learning teams.

In order to conduct reliability assessment, the development team must define proper reliability assessment methods, which must be aligned with project goals as defined in the *Data Understanding* phase [6]. Defining the expectations and goals of the reliability assessment process are necessary to establish a solid foundation on which the process can be built. The *Data Understanding* and *Data Preparation* phases involve thorough documentation of found feature characteristics, created boundaries and uncertainties such as binning limits, regressor accuracies and missing value handling in the dataset. While these tasks are not specific to reliability assessment, they are often neglected

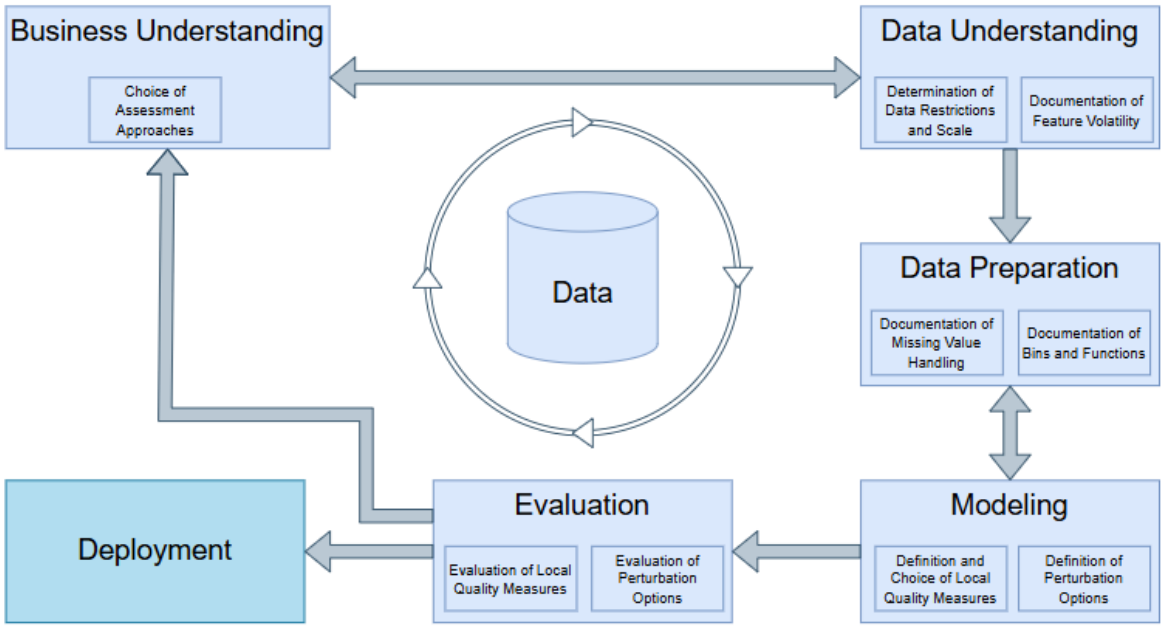


Figure 2.2: Adapted CRISP-DM process for reliability assessment. Adapted from [6].

by project teams [5]. For perturbation analysis specifically, these boundaries and uncertainties are the foundation on which *perturbation options* are built. Therefore, the reliability assessment reference process adds this particular focus on data documentation to CRISP-DM. During the *Modelling* phase, development teams process previously defined metrics and create methods for used perturbation options or other assessment approaches, grouped under the term *local quality measures* in Figure 2.2. Definitions of perturbation options then form the basis for perturbation and define the operations and parameters used in perturbation analysis. The *Evaluation* phase is used to evaluate not only the model's performance, but also the choice and definitions of perturbation options or parameters for other local quality measures. Lastly, during the *Deployment* phase, new predictions are evaluated according to the aforementioned assessment approaches in order to finally assess the reliability of individual prediction outcomes. This increases operator trust in individual predictions and allows identification of problematic cases.

2.2.1 Perturbation Options

Perturbation analysis is the focus point of this thesis. The perturbation options are a crucial component of perturbation analysis, as they define the operations applied to each feature that is to be manipulated. A perturbation option typically consists of the attributes (i) *option identifier*, (ii) *feature identifier*, (iii) *identified scale of feature*, (iv) *perturbation level*, (v) *generation algorithm* and (vi) *optional parameters needed for generation*. The *option identifier* (i) assigns a unique ID to each perturbation option that can be referenced when the impacts of perturbation are evaluated. The *feature identifier* (ii) and *feature scale* (iii) are feature-dependent characteristics and required to establish a connection between perturbation option and feature space. The (iv) *perturbation level* defines the criticality of an option and is explained in the following section. The *generation algorithm* (v) describes the alteration process for the corresponding feature. The *generation algorithm* combined with *perturbation parameters* (vi) specify the extent and method of feature perturbation. For example, given an input value of amplitude 100 that is to be perturbed by percentage with a perturbation parameter of 5% and step size of 1% transforms the input value

into a range of $[95, 96, 97, ..., 103, 104, 105]$. In summary, perturbation options serve as a blueprint for data generation, which is then input into predictors and further analysed.

2.2.2 Perturbation Levels

Each perturbation option is associated with a perturbation level, indicating the significance of a prediction change caused by altering input data as defined by the option. These levels are colour-coded and take on either of the levels *green*, *orange* or *red*. The levels are ranked by importance, with *red-level* options representing the most critical and *green* the least critical perturbation options. The levels are assigned based on findings from the *business understanding*, *data understanding* and *data preparation* phases of the reliability assessment process. Red levels are assigned to options for features whose actual physical value may not be known to the system at runtime. Sensor precisions, binned data structures or regressor functions are examples that would classify a perturbation option as a *red-level* option. For example, if an input value derived from a physical sensor is altered within the margins of its sensor's precision and this marginal manipulation is enough to change the prediction outcome, that particular input value might be close to internal model boundaries and is therefore considered unreliable. *Orange-level* options, like *red-level* options, are not expected to change within the defined margins. However, predictive changes caused by *orange-level* perturbations are deemed less critical. The decision to chose between *red* and *orange* levels could be based on the importance of the corresponding feature to the business case, or error margins not being as strictly defined as sensor tolerances for example. Green-level options are features for which the perturbation may be of interest to stakeholders, regardless of the expected predictive outcome. Generally, a green-level perturbation is expected to result in a change. For example, an ordinal feature with four possible values $[0, 1, 2, 3]$ might be classified as a green-level feature. A change in input from 2 to 1 might be enough to expect a change in prediction. Ultimately, the development team must determine suitable perturbation options according to previously documented findings during the reliability assessment process.

2.2.3 Perturbation Modes

Contrary to perturbation levels, perturbation modes are not feature-specific and instead define how the overarching perturbation approach is conducted. By limiting the number of perturbation options used or establishing operational order for the perturbation process based on perturbation levels and feature importance, the process can be controlled to be more computationally efficient when necessary. The perturbation process generates new input vectors for prediction based on the defined perturbation options. This causes the dataset to grow significantly for each additional perturbation option. Furthermore, combinations of perturbation options increases the amount of data vectors exponentially, leading to even larger input matrixes that are increasing the time and memory capacity needed to conduct perturbation. To address this, Staudinger et al. [6] introduced the three modes (i) *full*, (ii) *prioritized* and (iii) *selected*. Each mode defines the scope of the perturbation generation. The computationally most expensive mode is (i) *full*, in which all declared options are used for perturbation. The (ii) *prioritized* mode assigns a priority value to each perturbation option such that the highest-priority option is computed first, followed by options in descending order until either a certain number of values are generated or a time limit is reached. Lastly, the (iii) *selected* mode limits the number of options to a predefined subset, discarding

all options that have not been selected. Furthermore, computational expense can be adjusted by expressing the amount of maximum overlapping perturbation generations. Allowing multiple perturbation options to be active at the same time causes data complexity to increase exponentially with each additional option. In contrast, restricting perturbations to single columns increases data complexity linearly, which may already be sufficient to identify unreliable classification scenarios depending on the use case and the goals of the reliability assessment approach.

2.2.4 Execution and Evaluation

To perform perturbation, a given input vector is altered as specified by the perturbation modes, levels and options. To explain the perturbation process more concisely, a simplified showcase scenario is illustrated in this section. The option definitions for the scenario are shown in Table 2.1 while Table 2.2 shows the input vectors for this showcase scenario highlighted in grey. The generated vectors with the perturbed values are listed beneath the grey input vectors. The perturbed values according to the option definitions and the changed prediction outcomes are highlighted according to the corresponding option's level.

Identifier	T
Feature Name	temperature
Parameter	3
Level	Red
Generating Algorithm	Absolute Change
Identifier	W
Feature Name	weather
Parameter	-
Level	Green
Generating Algorithm	Exchange All
Identifier	RC
Feature Name	riskClass
Parameter	1
Level	Orange
Generating Algorithm	Absolute Change

Table 2.1: Exemplary perturbation option declarations.

The perturbation options defined in Table 2.1 include varying levels and parameters to provide insights into different possible perturbation option definitions and their impacts on the generated output, as seen in Table 2.2. The generating algorithms are referenced by a keyword that describes the algorithm used to perturb the original input vector. The logic of the algorithm can take on any form that is suited to the particular perturbation option, as identified in previous phases of the reliability assessment approach. For this showcase example, the generating algorithms are defined as 'Absolute Change', which alters the input value by an absolute amount up to the

maximum given by the parameter value, or 'Exchange All', which replaces the original value with all other possible values. The original input value describes a fictional scenario in which a predictor predicts if a flight will be arriving delayed at a given airport. The target variable is called *isDelay*. The *rowId* either indicates the original value, or references the option responsible for the alteration given by the option identifier, as seen in Table 2.1. For this exemplary scenario, two single column perturbation scenarios using mode *full* are shown. The temperature column contains cardinal values ranging from 0.0 to 30.0. The weather feature is nominal and indicates the weather conditions at the airport, with possible values of $['clear', 'cloudy', 'rainy']$. The feature *riskClass* is ordinal, representing a potential risk factor for flight delays with values ranging from 0 to 3. Table 2.2 illustrates the process of generating new input vectors based on the perturbation options. The two original input vectors are highlighted in grey. Value changes due to perturbation are highlighted based on the level of the perturbation option. In perturbed data vectors with differing prediction outcomes to the corresponding original vector's predicted target, the target variable is highlighted according to the colour of the connected perturbation option's

rowId	temperature	weather	riskClass	isDelay
original<1>	20	clear	0	no
T<1>	21	clear	0	no
T<1>	19	clear	0	no
T<1>	22	clear	0	no
T<1>	18	clear	0	yes
T<1>	23	clear	0	no
T<1>	17	clear	0	yes
W<1>	20	cloudy	0	no
W<1>	20	rainy	0	yes
RC<1>	20	clear	1	no
original<2>	10	rainy	2	yes
T<2>	11	rainy	0	yes
T<2>	9	rainy	0	yes
T<2>	12	rainy	0	yes
T<2>	8	rainy	0	yes
T<2>	13	rainy	0	yes
T<2>	7	rainy	0	yes
W<2>	10	cloudy	0	yes
W<2>	10	clear	0	yes
RC<2>	10	rainy	1	yes
RC<2>	10	rainy	3	yes

Table 2.2: Showcase of perturbation scenario.

level. In the first scenario identified by $rowId =' original < 1 >'$, the prediction changes due to the red-level perturbation option for the *temperature* feature when temperatures are lower than 19 degrees. Additionally, the green-level perturbation for the *weather* feature predicts a delayed flight due to rain. As a result of these changes, the original prediction would be marked as *alarmed* and classified as unreliable, since a red-level option was responsible for the change. The second scenario, indicated by $rowId =' original < 2 >'$, shows no changes in prediction and is therefore considered reliable. The showcased example also illustrates the increased data complexity caused by the perturbation approach. In the showcased example seen in Table 2.2, the data matrix given to the predictor grows by up to 10 entries for each original data vector. In the example, in total of 19 new data entries are generated and classified over all options and input vectors. For such single-column perturbation scenarios, data complexity grows linearly for each additional perturbation option and data vector. In a multi-column perturbation approach, this example would expand from 10 altered data rows per original entry to 63 rows, calculated as $n_T \cdot n_W \cdot n_{RC} = 7 \cdot 3 \cdot 3$ where n_T, n_W and n_RC are the number of alterations per *temperature*, *weather* and *riskClass* respectively. Especially for resource intensive machine learning models, this could lead to runtime issues, making it necessary to carefully assess the perturbation modes and limit the number of column combinations to the minimum.

2.3 Related Work

This thesis uses the reliability assessment process of Staudinger et al. [6] to evaluate possible benefits and challenges associated to the proposed process on the aviation domain. To evaluate the reliability, Staudinger et al. proposed a method called *perturbation analysis* in which marginal alterations in input vectors' impact on prediction outcomes is analysed. Furthermore, this thesis is strongly related to Bardach et al. [11] which classified delay risk classes on flights arriving in Hartsfield-Jackson Airport in Atlanta United States and Vienna International Airport. They used flight data, meteorological air reports, notice to airmen and aircraft data to train the classifiers, reaching 82.5% accuracy on the most important risk class identified. This thesis, uses the same data combination and time frame as Bardach et al. on which the reliability assessment approach defined by Staudinger et al. is performed on.

Flight delay prediction is a field critical to modern-day flight operation and is a widely adapted use case for machine learning applications. Thiagarajan et al. [12] used a two stage predictive model to predict flight delays of domestic passenger carrier flights. The first stage consists of a Gradient Boosting Classifier predicting a binary class representing the occurrence of a delay, with the second stage consisting of an Extra-Tree Regressor model to predict carrier delays in minutes. For the flight delay prediction project, domestic flight data from 2012 to 2016 consisting of weather and carrier on-time performance data was collected and used for training. The first stage achieved an accuracy of 94.35% on the binary classification task, while the regression task showed a Root Mean Squared Error of 26.36. Vo et al. [13] classified flight delays in a multi class classification scenario where a flight is classified in either one of the classes being *on-time*, *delays of more than 30 minutes* and *delays less than 30 minutes*. By using temporal flight data like departure delay, time of day and day of week as well as flight characteristics such as origin and destination airport and distance between each, they achieved a test accuracy of 64.86% with a decision tree classifier. Since aviation handles large amount of data, their focus was set on

creating a practical prediction system capable of handling large datasets by incorporating the Apache services Cassandra, Spark and Kafka into the machine learning pipeline. Jiang et al. [14] predicted flight delay classes in 5 degrees of delay types ranging from no delay with flight delays less than 15 minutes to extensive delays capturing flights that are delayed by more than 240 minutes with brackets of varying sizes in between. By combining weather information with flight information, they were capable of training a Random Forest Classifier with 80.21% accuracy and a Multilayer Perceptron with 89.07% accuracy over all classes. Nigam and Govinda [15] similarly uses a combination of weather data and flight on-time performance on a Linear Regression model, achieving 80.6% accuracy over a binary classification task identifying on-time or delayed flights over 70 airports in the United States. Hendrickx et al. [16] uses exclusively flight information to predict delays with a binary classification scenario, reaching 76.6% accuracy with Random Forest Classifier and Multilayer Perceptron models. Particular focus on oversampling techniques and their influence on the model training process has been studied throughout their work. Gui et al. [17] tried to establish a LSTM model using the radio transponders of aircraft to introduce a traffic metric based on number of flights within the same corridor between airports in addition to flight and weather data. The LSTM did suffer from over-fitting due to a limited training dataset, which was alleviated by using a Random Forest Classifier capable of differentiating strong delays of more than 60 minutes from lower delays, achieving an accuracy of 90.2%. Moreira et al. [18] built a neural net with back propagation trained on flight and weather data regarding multiple airports located in Brasil, reaching an accuracy of 73.81% on a binary classification task with SMOTE oversampling of the minority classes.

With establishing machine learning models for sensitive tasks comes the need to assess the reliability of individual predictions produced by corresponding models. Reliability assessment of individual predictions is an open research field and gains importance with increasing numbers of deployed machine learning models. Bosnić and Kononenko [19] investigated various methods for reliability assessment of individual predictions, covering methods using sensitivity analysis, bagging models, local performance metrics and local density metrics. They evaluated these reliability measuring methods by testing against various blackbox regression models. To evaluate the performance of the respective approaches, correlation coefficients between corresponding reliability assessment metrics and signed prediction errors are compared. Zhang and Bose [7] introduced a quantitative metric R(x) to assess individual prediction reliability by observing the differences in predicted and true target values within a fixed neighbourhood around the to be checked input vector and calculating the proportions of neighbours that violate a given threshold within their predicted differences. Furthermore, they proposed an information gain-driven threshold that indicates whether a resulting R(x) is deemed unreliable for a given prediction. Since calculating deviations in prediction outcomes for a given neighbourhood is computationally expensive, especially for models with high numbers of features, they introduced a second stage to their prediction problem to predict R(x) values for a given model based on the input vector. This allows for faster calculations for additional vectors during deployment stages.

3

Model Implementation

In this chapter, we describe the intricacies of the CRISP-DM process concerning the model building phases for this thesis. Each phase will be documented step-by-step to give a comprehensive overview of the development decisions and interesting findings. Furthermore, each phase includes the additional steps required for the perturbation process, which are essential for the analysis in the next chapter.
The chapter follows the structure outlines in the CRISP-DM guidelines [8]. In Section 3.1, we examine the complexities of the aviation industry and the challenges and risks that are present for stakeholders connected to the industry. Section 3.2 explores the sources, quality and statistical properties of the available data for this project. Section 3.3 explains the necessary data cleaning and processing steps, such that the quality of the input data is sufficient for the model building phase. Section 3.4 establishes an overview of used machine learning techniques and goes into detail for the model selection processes that have been used in this project. Finally, Section 3.5 presents the classification results and evaluates the models based on selected performance metrics.

3.1 Business Understanding

The aviation industry relies heavily on the punctuality of arrivals and departures to maintain operational efficiency. Precise scheduling is critical for coordinating connecting services, optimizing business functions, and reducing operational costs. Even minor delays can trigger cascading effects throughout the operational chain, adding delays to a network of interconnected flights [20]. Airlines that depend on efficient operations to minimize standing costs and maximise aircraft usage are especially vulnerable to the financial burdens of unforeseen delays [4]. It is estimated that in the year 2019 delayed carriers cost airlines in the US around $33 billion USD in direct costs, lost demands and other indirect costs [21]. Furthermore, delays cause a decrease in customers trust in airlines, additionally increasing financial pressure on operators [4]. Unforeseen early arrivals may also create congestions at airports, leading to cascading delays for subsequent flights, while also indicating missed potential in airlines' fleet management potentials [22]. Therefore, both early and late arrivals contribute to costly standing times and are capable of disrupting gate scheduling tasks, which negatively impacts the capabilities for efficiently scheduling departures and arrivals of airports [23]. To mitigate these disruptions to day-to-day services, predictive systems may be capable of identifying delayed flights in a timely manner. For airlines, such systems provide a competitive edge by enabling cost-efficient adjustments to schedules and resource allocations. Airports also

benefit from delay prediction systems, since early recognition of unpunctual arrivals allows for better gate scheduling, optimized staff management and reduced downtime at busy terminals. Looking into these characteristics regarding flight delays, the main stakeholders are passengers, airport operators and airlines. Passengers have the interest of arriving at the destination with maximum comfort and efficiency. Secondly, airport operators benefit of better organizational capabilities, improving customer comfort and reputation for airlines. Lastly, airlines are capable of minimizing costs and leveraging strategic and administrative advantages in respect to their competition. For the aforementioned stakeholders, accurate classification of on-time arrivals is therefore a top priority. High precision ensures that stakeholders do not mistakenly prepare for flights arriving earlier or later than anticipated.
To enhance the confidence in decisions based on the predictive system, the perturbation approach will be followed to assess the reliability of individual predictions. By identifying unreliable predictions, operators can manually validate results using their operational experience, supporting model decision-making and reducing the cost of misallocating resources. Since delays of individual carriers often propagate to subsequent flights of the same carrier, earlier delays within an operational day are critical for accurate classification. In the scope of this thesis, a use-case has been defined such that the machine learning application predicts whether a delay or early arrival occurs at the time of take-off from the origin airport. This use-case definition ensures that the system knows the true departure time at the time a prediction is made, making the departure delay feature suitable for the classification task, covering one of the most critical attributes for delay prediction [24]. In addition to having departure delays known at the time of departure, weather data can also be collected in real time making assertions regarding weather situations possible without needing additional model predictions for weather attributes in the data set.

3.2 Data Understanding

The data relevant to the project comes from four distinct sources, which were integrated into a single table to form the input matrix for subsequent analysis following the CRISP-DM methodology. The datasets are analysed based on their statistical properties on a column-by-column basis. Furthermore, key design decisions and findings essential for the later perturbation analysis are outlined in this section. Based on the supervising institute's recent experience with parts of the dataset, as well as its general availability and documentation, this thesis uses the dataset on domestic passenger flights within the United States with a particular focus on flights to Hartsfield-Jackson Atlanta International Airport as described later. The following sections present the findings for (i) Flight Data, (ii) Aircraft Data, (iii) Meteorological Air Report Data (METAR), and (iv) Notice to Airmen (NOTAM) entries, respectively. Initially, data from 2016 to 2017 was selected, as the work of Bardach et al. [11] investigated the same time frame, which allows building on experiences and findings made within the scope of their work regarding the dataset. Furthermore, the idea of adding flights from 2020 onward was discarded regardless due to the COVID-19 pandemic and its disruptive impact on the aviation industry, which led to significant deviations from typical flight patterns [2]. Additionally, NOTAM data was only available for 2017, which became the limiting factor that ultimately lead to the decision to focus solely on flights departing in 2017 for the later stages of the thesis.

3.2.1 Flight Data

Flight data was obtained from the Bureau of Transportation Statistics (BTS) in the United States. The BTS stores reports, statistics, and datasets on various topics covering all domains of transportation related to the US. For this thesis, the dataset used was taken from the aviation category and is titled 'Reporting Carrier On-Time Performance (1987-present)' [25]. The data is maintained by the Office of Airline Information within the BTS and contains information about individual flights operated by certified US-carriers [26]. For this thesis, we used the same initial attribute-set as Bardach et al. [11] which neglected redundant information of airport locations while also discarding more detailed diverted aircraft information and gate return information. The dataset provides time-performance details of flights, including actual and scheduled departure and arrival times. It also includes cancellation indicators, as well as causes and magnitudes of delays.

Dataset Overview

The downloaded dataset including flight data from 01.01.2016 to 31.12.2017 holds $11.292.279$ reported flights with 28 attributes per recorded flight. Table 3.1 shows the data attributes in their unprocessed form with data types, ranges, null value counts and example values, to give a broad overview of the data structure. Meta information like dates, identification numbers, cancellation codes and airport codes are represented as strings while almost all attributes regarding time data are represented either as integers or floats which do not use the floating-point capability of the data type.

Date and Time Data

Almost all time-sensitive data attributes capture the amount of delay in minutes represented by all columns with the suffix *'delay'*. Other entries capturing time objects are the columns *'dep_time'*, *'crs_dep_time'*, *'arr_time'*, *'crs_arr_time'*, *'wheels_off'* and the date column *'fl_date'*. The aforementioned time columns represent the time of day as displayed by a digital clock. The value indicates the time of day at which a corresponding event of a given flight took place. Since these date columns are stored as integers, transforming each value into a datetime object is necessary to make further processing possible. The date-column *'fl_date'* uses a string representation for the date and represents the date of departure as planned in the Computer Reservation System (CRS). Combining the date-columns with each of the time-columns yields datetime objects, which are then usable for further calculations and presentations. The arrival delay forms the target variable for the machine learning task and is calculated by the difference of actual arrival time and arrival time as defined in the CRS.

Null Values

Most null occurrences within the flight dataset indicate cancelled or diverted flights due to the absence of measurements on such events. All time-sensitive attributes besides the CRS-attributes equal *'null'* on both diverted and cancelled flights. However, the tail number of a flight is *null*, only when the flight was cancelled. All CRS-attributes are non-null, since every reported flight must already have a planned departure and arrival time for it to be registered in the system. Nevertheless, the attribute *'crs_elapsed_time'* includes 13 null values, which most likely stem from missing inputs even though the dataset documentation does not mention

Table 3.1: Attribute properties of initial flight dataset.

Name	Type	Range	# Null	Example
fl_date	string	char(10-10)	0	2016-01-01
op_unique_carrier	string	char(2-2)	0	WN
tail_num	string	char (5.0-6.0)	25403	N7728D
op_carruer_fl_num	int	[1;8402]	0	1596
origin	string	char(3-3)	0	ABQ
dest	string	char(3-3)	0	BWI
crs_dep_time	int	[1;2359]	0	1440
dep_time	float	[1.0;2400.0]	143764	1505.0
dep_delay	float	[-234.0;2755.0]	143799	25.0
taxi_out	float	[0.0;186.0]	147563	7.0
wheels_off	float	[1.0;2400.0]	147559	1512.0
wheels_on	float	[1.0;2400.0]	152518	2014.0
taxi_in	float	[0.0;414.0]	152518	4.0
crs_arr_time	int	[1;2400]	0	2015
arr_time	float	[1.0;2400.0]	152518	2018.0
arr_delay	float	[-238.0;2189.0]	174724	3.0
crs_elapsed_time	float	[1.0;718.0]	13	215.0
actual_elapsed_time	float	[14.0;784.0]	174724	193.0
air_time	float	[4.0;723.0]	174724	182.0
distance	float	[25.0;4983.0]	0	1670.0
cancelled	float	[0.0;1.0]	0	0.0
cancellation_code	string	char(1.0-1.0)	11143725	A
diverted	float	[0.0;1.0]	0	0.0
carrier_delay	float	[0.0;2142.0]	9298567	9.0
weather_delay	float	[0.0;1934.0]	9298567	0.0
nas_delay	float	[0.0;1605.0]	9298567	1.0
security_delay	float	[0.0;827.0]	9298567	0.0
late_aircraft_delay	float	[0.0;1756.0]	9298567	9.0

the possibility of *'null'* values for *'crs_elapsed_time'*. Furthermore, attributes that consist of more than 80% empty fields are *'cancellation_code'* and the delay columns *'carrier_delay'*, *'weather_delay'*, *'nas_delay'*, *'security_delay'* and *'late_aircraft_delay'*. Missing values for delay fields are indicating flights for which the values were not reported by flight operators. Lastly, the attribute *'cancellation_code'* represents the reasoning behind flight cancellations according to the logic of $A \rightarrow$ carrier, $B \rightarrow$ weather, $C \rightarrow$ national air system and $D \rightarrow$ security. Empty fields describe the absence of a cancellation event, allowing the flight to be processed as planned.

3.2.2 Aircraft Data

Aircraft data is retrieved from the Federal Aviation Administration's aircraft characteristics database [27]. The dataset is updated regularly and includes 2764 different aircraft types. The dataset provides dimensional metrics such as wingspan, tail height and length of specific aircraft models as well as configuration metrics like maximum allowed take off weight, number of engines and types of winglets used for various aircraft models. In total, the dataset consists of 25 attributes. Bardach et al. [11] manually enriched this aircraft database with additional metrics and filled in missing entries for popular aircraft models within the scope of their carrier on-time delay classification thesis, which improved the original dataset's quality. Since the additional data fields are useful to the flight delay classification task within this thesis, the aircraft dataset from Bardach et al. was adapted. Although the data quality was improved through the additional data entries, the dataset still contains 33.565 missing entries, amounting to 49% missing values in the dataset. However, the missing entries mostly concern less popular aircraft models, as explained in Section 3.3.

Dataset Overview

Table 3.2 shows all features of the dataset with their data types and ranges. The example column shows entries regarding the popular Boeing model *'737-800'*. All dimensional attributes from *'Wingspan'* to *'MGW'* are given in the imperial unit feat, while *'Parking Area'* is derived by multiplying *'Wingspan'* and *'Length'* therefore showing its value in square feet. Other remaining cardinal features are the weight features *'MTOW'* and *'Max Taxi'* which are given in pounds, as well as the speed feature *'Approach Speed'* which is given in knots. Besides the described cardinal features, the dataset includes categorical and ordinal features describing various classification IDs for the aircraft in *'AAC'* (Aircraft Approach Class), *'ADG'* (Aircraft Design Group) and *'TDG'* (Taxiway Design Group) which are indicators for dimensional characteristics of an aircraft and therefore correlate strongly with corresponding dimensional features. The *'Physical Class'* feature holds either of the two possible engine configurations *'Jet'* or *'Turboprop'* with *'Engines'* giving the number of engines mounted on the aircraft.

Null Values

With 19 of the 25 features including more than 1500 *null* entries, the dataset contains 59.9% values holding *'null'* or *'tbd'*. Because of this suboptimal data quality for machine learning tasks, subsequent data preparation steps are critical for the usability of the dataset. Although the general data quality is not optimal, prominent aircraft models such as the *'Boeing 737-800'*, shown in the example column, are represented almost completely with only the two hand-picked data attributes being empty. However, due to insignificance to the machine learning task, these particular attributes are later dropped anyway, making most popular aircraft entries complete. With this perspective, the aircraft dataset will be considered to the machine learning task regardless of the overall bad data quality, since most flight operations will be conducted using popular aircraft models as carriers, resulting in a higher percentage of valid data entries in the final integrated dataset used for training.

Table 3.2: Attribute properties of initial aircraft dataset.

Name	Type	Range	# Null	Example
Date Completed	string	char(3-11)	2054	2016-Jun-17
Manufacturer	string	char(3-83)	0	Boeing
Model	string	char(1-144)	0	737-800
Physical Class (Engine)	string	char(3-19)	1831	Jet
Engines	int	[0;4]	1956	2
AAC	string	char(1-8)	0	D
ADG	string	char(1-20)	0	III
TDG	string	char(1-7)	2038	3
Approach Speed	float	[30.0;230.0]	1911	142.0
Wingtip Config	string	char(3-21)	2023	no winglets
Wingspan	float	[19.25;261.65]	1906	112.58
Length	float	[17.33;250.17]	2025	129.5
Tail Height	float	[3.67;79.3]	1919	41.42
Wheelbase	float	[5.42;107.87]	2303	51.17
Cockpit to Main Gear	float	[5.42;121.59]	2303	56.42
MGW Outer to Outer	float	[6.04;47.05]	2302	22.96
MTOW	int	[850;1300000]	1916	174200
Max Taxi	int	[850;1305000]	2164	174900
Main Gear Config	string	char(1-8)	1969	D
ICAO Code	string	char(2-16)	537	B738
Wake Category	string	char(1-3)	629	M
ATCT Weight Class	string	char(3-20)	2172	Large Jet Eqpt
Years Manufactured	string	char(3-30)	2675	tbd
Note	string	char(3-85)	2742	tbd
Parking Area	int	[341;62435]	2025	14580

3.2.3 METAR Data

Weather data plays an important role in the efficiency and safety of the aviation industry. A common format used to provide weather data is the Meteorological Aerodrome Report (METAR). METAR provides standardized and structured data to decision makers in real-time to allow for better flight planning and airport management. The dataset used for this thesis was provided by the Iowa Environmental Mesonet (IEM) which collects sensor and human-observed data through weather stations around the world organized by the Iowa State University [28]. Airports in the United States are equipped with the Automated Surface Observing System (ASOS) which was used by the IEM to fetch the specific airport METARs needed for this thesis. Additionally, this particular dataset focuses on METARs concerning the Hartsfield-Jackson Atlanta International Airport (KATL), spanning the time-frame from January 2016 to December 2017 to match the flight dataset described in Section 3.2.1. For better data understanding, the ASOS User Guide

[29] provides a documentation explaining the processes and techniques used for data collection and is further used to deduce limitations of attributes and define perturbation scopes.

Table 3.3: Attribute properties of initial METAR dataset.

Name	Type	Range	# Null	Example
tmpf	float	[15.1;98.96]	172637	51.98
dwpf	float	[1.0;77.0]	172639	42.08
relh	float	[8.99;100.0]	172669	68.86
drct	int	[0;360]	5400	320
sknt	int	[0;39]	4879	10
p01i	float	[0.0;1.84]	168570	0.0
alti	float	[29.15;30.7]	31	30.19
mslp	float	[987.1;1041.2]	175544	1022.5
vsby	float	[0.12;10.0]	0	10.0
gust	int	[13;53]	187790	16
skyc1	string	char(3-3)	7644	FEW
skyc2	string	char(3-3)	146844	SCT
skyc3	string	char(3-3)	172045	BKN
skyc4	string	char(3-3)	191156	BKN
skyl1	int	[0;30000]	96692	10000
skyl2	int	[400;30000]	146844	15000
skyl3	int	[800;30000]	172045	20000
skyl4	int	[13000;30000]	191156	25000
wxcodes	string	char(2-14)	175124	-RA
ice_accretion_1hr	float	[0.01;0.07]	193022	0.03
ice_accretion_3hr	float	[0.07;0.13]	193031	0.13
ice_accretion_6hr	float	[0.07;0.24]	193031	0.24
peak_wind_gust	int	[26;56]	192464	26
peak_wind_drct	int	[10;360]	192464	320
peak_wind_time	string	char(3-16)	192464	2016-01-04 19:10
feel	float	[1.11;104.55]	172669	51.98
metar	string	char(3-225)	1702	KATL 010052Z 32010KT 10SM FEW100 SCT150 BKN200 BKN250 1106 A3019 RMK AO2 SLP225 T01110056
date	string	char(10-10)	0	2016-01-01
time	string	char(5-5)	0	00:52

Dataset Overview

Table 3.3 outlines the attributes of the METAR dataset, showing the variety of information provided by each data record. The dataset includes information about sensors stationed around airports. Attributes such as air temperature *'tempf'*, dew point temperature *'dwpf'*, wind direction *'drct'*, wind speed *'sknt'*, precipitation *'p01i'*, visibility *'vsby'*, air pressure altimeter *'alti'* and the sky level altitude attributes *'skyl1'* to *'skyl4'* are collected by the aforementioned sensors at set times. The rate at which the sensor values are read differ between the various measurements, as documented in the ASOS User Guide [29]. Besides sensor data, the METAR provides the user with derived data such as relative humidity *'relh'* which is based on surrounding temperature and dew point temperature measurements, while the sea level pressure *'mslp'* is dependent on altimeter measurements and corresponding altitude above sea level at the sensor's location. Additionally, METAR provides observatory data with the sky level coverage attributes *'skyc1'* to *'skyc4'* and the weather event attribute provided by *'wxcodes'*. Both of these attribute categories are encoded and provide information about cloud density or weather events like rainfall and hail, among others. Lastly, METAR provides information about peak wind occurrences, directions and speeds as well as various levels of ice accretion within a given time interval in the data features *'ice_accretion_1hr'* to *'ice_accretion_6hr'*. The combination of attributes allows for a precise representation of various weather scenarios. In addition to the mentioned attributes, the data column *'metar'* provides the user with a raw METAR as received by connected systems, as seen in the example column of Table 3.3. Features are mostly collected at different times of the day, leading to high feature sparsity in the dataset's rows. Therefore, unlike in previous tables showing data characteristics in this thesis, the values in the *'Example'* column of Table 3.3 are not connected to each other and are selected as the first value for each attribute that is not represented by a missing value.

Null Values

Contrary to the flight dataset, the occurrence of *null* within the dataset does not infer much meaning in the METAR dataset besides an indication for missing values. Many of these missing values are due to the different time intervals that ASOS uses for its measurements. Information regarding wind situation, visibility and air pressure are updated in 5 minute intervals while other measurements are updated hourly. This results in around 80% *null* entries for most records. Additionally, sensor downtime due to maintenance or obstruction further increases the portion of *null* entries within the dataset.

Weather Events

METAR includes textually encoded weather events to inform readers of ongoing meteorological scenarios. It may additionally include an indicator for the scenario intensity by appending '+' or '-' prefixes in front of the event code. Table 3.4 depicts the codes of such weather events and their corresponding meaning according to the National Weather Service of the United States [30]. For example, the event indicators '-RA', 'RA' or '+RA' imply the occurrence of slight rain, regular rain or heavy rain respectively at the corresponding time interval as defined in the METAR. These events are collected in a semi-automatic fashion, with some events being added manually while others are collected by sensor data according to the ASOS User Guide [29].

Table 3.4: METAR weather event code map.

TS	Thunderstorm	FG	Fog
RA	Rain	VA	Volcanic Ashe
DZ	Drizzle	BR	Mist
SN	Snow	HZ	Haze
SG	Snow Grains	DU	Dust
IC	Ice Pallets	FU	Smoke
GR	Hail	SA	Sand
GS	Small Hail	PY	Spray

3.2.4 NOTAM Data

Notice to Airmen (NOTAM) are short semi-structured messages, sent to terminal personal or aircraft crew to inform them of specific scenarios or challenges on the airport, the flight route and the aircraft. They are critical for aviation safety and include a wide variety of messages. Route changes, runway obstructions and status reports are some examples of the structured portions of NOTAM contents. Additionally, NOTAMs can include messages in text format, informing airport personnel about specific events. Due to the importance of NOTAMs for the aviation industry, this thesis incorporates NOTAMs to recognize challenges of given routes which could possibly lead to on-route-delays. Specifically, information about runway availability is processed for the flight delay prediction task in this thesis.

NOTAM data is stored and made available for download by the Federal Aviation Administration [31]. The data specifically used for this thesis was taken from sources provided to the supervising institute during the work of Bardach et al. [11] which included the relevant time-frame and some important features already extracted from the raw form of NOTAM reports. The collected dataset contains 16.163 NOTAM reports sent from Atlanta Hartsfield-Jackson Airport (KATL), spanning the time-frame from December 2016 to January 2018. NOTAMs are similar to METARs such that both are transported in semi-structured text and inform industry members about current events. Contrary to METAR data, the structure of NOTAM reports deviates strongly based on the type of report and the information that is provided by it. This higher variety between reports makes processing of NOTAM strings more challenging and vital for further operations on the dataset.

Table 3.5 shows three NOTAM representations as they occur in the obtained dataset. The NOTAM reports are marked from A to E breaking up the reports in their structured cells. Although NOTAM reports must preserve their overall cell structure, as shown in Table 3.5, the report message (D) varies strongly between NOTAMs and might include (i) surface designators, (ii) surface segments, (iii) facility components, (iv) location descriptions, (v) conditions and (vi) additional remarks for further information in a more or less freely chosen combination. The first example shown in Table 3.5 indicates a runway report concerning runway *09L*. The field condition (FICON) of the runway is described as 100% wet, with a code of '5' for all thirds of the runway. Codes range from 0 to 5 where lower numbers indicate higher danger [32]. The second NOTAM shown in Table 3.5 shows a runway report concerning runways *10* and *28*. However, the NOTAM message does not include a FICON report, but informs the reader about the closure of described runways

in the corresponding timeframe by attaching the keyword '*CLSD*'. The third NOTAM example given in Table 3.5 was chosen to show an example of report types besides runway reports. This case informs the reader about taxiway '*N*' which is partly closed east of taxiway '*T*'. As indicated by these examples, NOTAM reports are capable of informing the reader about various scenarios influencing decisions of operators in the field. From runway situations, airport management to special events taking place near aviation infrastructure, NOTAMs are integral for organization and decision-making processes.

Table 3.5: Showcase of NOTAM reports in the dataset.

Example Notams	
1	**(A)**!ATL 01/006 **(B)**ATL **(C)**RWY **(D)**09L FICON 5/5/5 100 PRCT WET OBSERVED AT 1701011728. **(E)**1701011728-1701011728
2	**(A)**!ATL 01/024 **(B)**ATL **(C)**RWY **(D)**10/28 CLSD **(E)**1701050430-1701051130
3	**(A)**!ATL 01/088 **(B)**ATL **(C)**TWY **(D)**N EAST OF TWY T CLSD **(E)**1701071355-1701071600

Part ID	**Explanation**
A	Reporting facility and ID
B	Broad location of occurrence
C	Keyword indicating type of report
D	Report message.
E	Effective and expiration dates (YYMMDDhhmm)

3.3 Data Preparation

Data preparation is the third phase within the CRISP-DM process. In this phase, the datasets are processed such that they are suited for the selected machine learning algorithm. This section describes the steps taken, with which the various datasets shown in Section 3.2 are transformed and combined into a single data table used for aircraft delay classification. Each data source has its individual characteristics and therefore possesses special preprocessing needs, which are being explained in separate sections. Then, the data integration section shows the process of combining useful data entries and features into one data matrix used for the later classification task.

Because of data sparsity, noisy data or suboptimal compositions, data quality is often not sufficient for machine learning tasks immediately after data gathering. Depending on the ML algorithm used, datasets must be curated to enable functioning machine learning processes that are capable of achieving set ML goals. The gathered data for the carrier delay classification task adopted in this thesis shows no exception for suboptimal data quality. Therefore, standard processes like handling empty values, removing data columns and rows with insufficient information are used on all subsets of the gathered data. In most cases, these processes are not explored in detail within the scope of this section unless they influence the generation of perturbation options explained in Chapter 4 or go beyond basic aggregation or deletion methods to improve data quality.

3.3.1 Processing the Flight Dataset

The flight dataset contains information about scheduled individual domestic flights within the United States. As shown in Figure 3.1, the Hartsfield-Jackson Airport in Atlanta (ATL) is the most frequented airport for domestic flights within the dataset and allows for the highest number of entries to be fed into the machine learning algorithm. Furthermore, in 2017 ATL was the busiest airport by departing and arriving passenger numbers globally [33]. Due to this significance of the airport and the focus on a single airport to keep the ML task minimal, ATL was chosen as target airport, which results in the elimination of all flight entries not having ATL as destination airport.

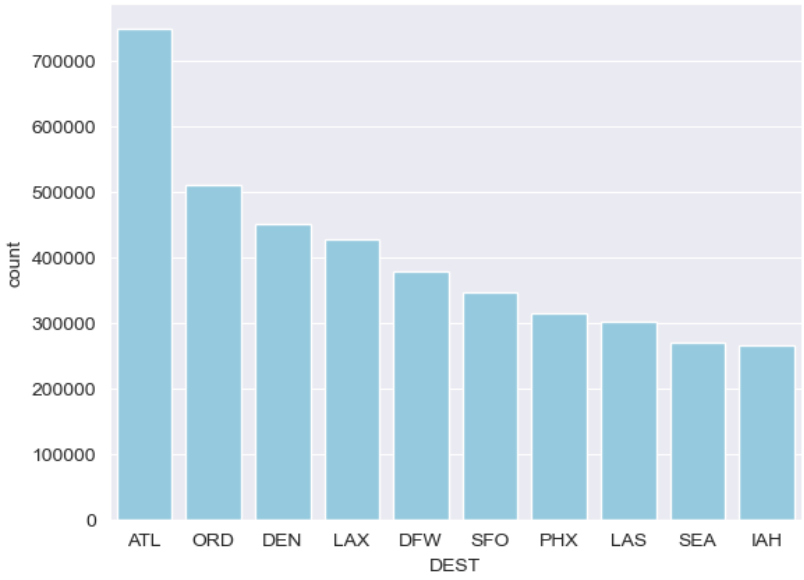


Figure 3.1: Countplot of the ten most frequented destination airports.

Generation of Timestamps

As shown in Section 3.3.1, the dataset provided by the BTS stores date and time data in separate feature columns. For a given flight, data regarding date of departure is represented as string formatted according to the expression 'YYYY-MM-DD' while time data is stored as string formatted by 'HH:MM'. Since *'fl_date'* is the only date indicator for each data row, the procedure of generating timestamp objects for departures and arrivals differed slightly. For departures, the timestamp is created by simply combining the features *'fl_date'* and *'dep_time'*. Due to flights possibly operating through midnight, the arrival timestamp created by combining *'fl_date'* with *'arr_time'* needs to account for a carry-over day if the calculated arrival timestamp is smaller than or equal to the departure timestamp. Since ATL is located in GMT-4 which is the easternmost time zone of the United States, only positive time zone deviations are possible, making negative carry-over days impossible for domestic flights. Moreover, the highest flight duration accounting for time zone influences in addition to air-bound travel time occurs for departures from *Daniel K. Inouye International Airport* in Honolulu, Hawaii. With a flight duration of roughly eight hours and 45 minutes and a time zone difference of six hours, the total possible time difference between departure and arrival is 14 hours and 45 minutes, making a second carry-over day impossible for domestic flights. Therefore, simply passing over a day when needed is sufficient to find the correct day of arrival. Intercontinental flights would need a more sophisticated solution for this problem.

Delay Attributes

The dataset includes features indicating reasons for delays. As mentioned in Section 3.3.1, these are rarely filled in by operators and are therefore discarded from the dataset. Additionally, features indicating more detailed time events like *'taxi_in'*, *'taxi_out'*, *'wheels_on'* and *'wheels_off'* have been dropped. These features exist in the *'dep_delay'* feature in an aggregated form, making the former features redundant. Although other works referred to in the related work chapter include the more detailed features in addition to the departure delay estimates or absolute values, experiments showed no benefit with earlier models, guiding the decision to drop the features.

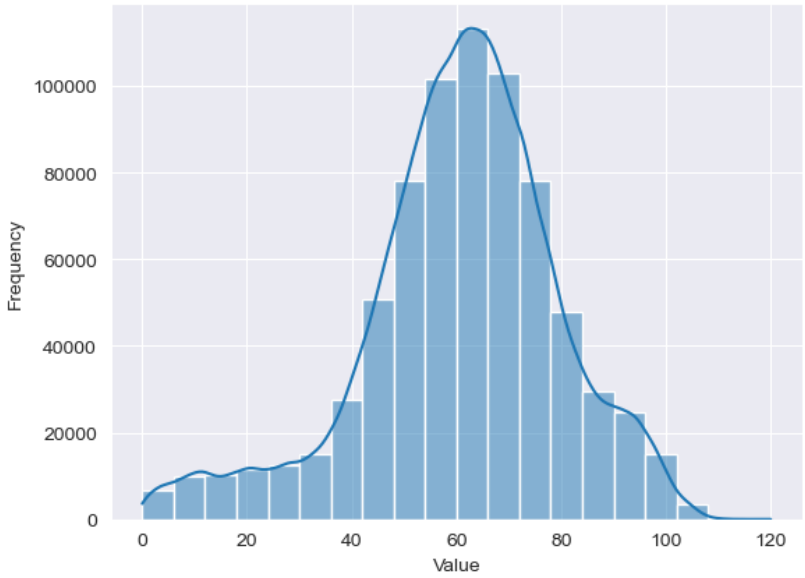


Figure 3.2: Distribution of number of flights arriving before a given flight in an 1 hour time-frame.

Traffic Volume

Congestions on routes, runways and taxiways are often negatively impacting punctuality of carriers. Often unforeseen weather events, organisational mistakes and problems with airport infrastructure can lead to increases in departure and arrival delays [4]. Therefore, a feature representing traffic on arrival airport is valuable for carrier delay prediction [34]. To manipulate the dataset such that it represents traffic information for each arriving flight, the number of flights arriving within a time span of one hour before the planned arrival according to CRS times is calculated. The CRS arrival times are used in favour of the actual arrival times, since a deployed model would not have information about actual times at runtime. Hu et al. [35] used the number of flights departing at a given airport as traffic identifier, while Zhang et al. [36] used a more complex indicator taking trajectory data of carriers into account. For this thesis, the number of flights arriving within one hour of the planned arrival time is calculated as a metric for traffic volume. With a higher traffic volume at the time of arrival, the risk of congestions increases, justifying the features' existence in training data. The time frame of one full hour was chosen for simplicity of calculation as well as better understandability of the feature. The resulting distribution with the chosen time frame over all flights looks as shown in Figure 3.2, with $std = 18.96$; $\mu = 60.63$ and the data range $[0; 120]$. Noteworthy, the maximum number of previous flights arriving one hour before a given flight is 120, which might indicate the limit to the arrival capacity of the airport.

Airport and Operator Indicators

Of all categorical features in the dataset, only two hold information about the flight's characteristics, beside identification. These features are codes for origin airports and flight operators relevant to the corresponding flight. To obtain trainable information on these datasets for the delay classification problem, the median arrival delay recorded in the previous year is inserted in place of the codes. Table 3.3 shows the frequency and corresponding median delays for flight operators in 2016. Since this thesis does not include NOTAM data from before 2017, we can safely use the 2016 delays as baseline for these features without inducing information bleeding. Since the flight operator Alaska Airlines (AS) hardly operated flights to ATL in comparison to the other airlines, the median value of all median flight operators was used to reduce the impact of outliers within the AS flights. The same process was conducted for the origin feature. Due to the high number of origin airports in the dataset and the insignificance of some, the 0.1 quantile was chosen as a threshold, eliminating all flights with frequencies beneath the threshold and inserting median delays over all other airports for these less significant airports.

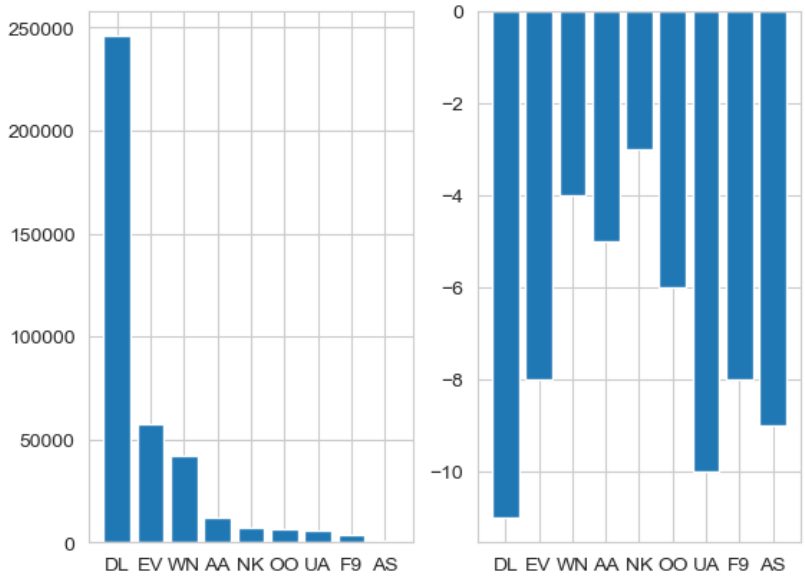


Figure 3.3: Count plot (left) and median delays (right) of flight operators in 2016.

3.3.2 Processing Aircraft Data

The flight dataset includes the tail numbers for each flight that has been carried out in the previously explained time frame. Tail numbers of planes are similar to registration numbers on personal vehicles, such that they might change whenever a given aircraft is transferred to other operators. Although re-registration rarely occurs, mapping correct tail numbers to the aircraft type at the time of flight operation is necessary to identify the aircraft model carrying out the route. The site *airsfleet.net* displays information about aircraft types for given tail numbers and was therefore used to gather the corresponding aircraft models. After scraping this information, out of the possible 4228 unique tail numbers found in the dataset, 3267 unique carriers which targeted ATL from 2016-2017 were identified. The scraped dataset includes entries about the aircraft model as a space delimited string in the form of *'{manufacturer} {model}'*. Since some manufacturers like *'McDonnell Douglas'* or *'Acro Sport'* already have a space character within their name, the manufacturer recognition has to be done manually to avoid false splitting operations. Since only

20 different aircraft types are located within the 3267 identified aircraft, which all are related to the biggest aviation companies in the United States, splitting multi-word names manually was a feasible task.

Table 3.6: Dimensions of various Boeing 737 models

Model	Approach Speed	Wingspan	Length	MTOW
737-100	128	93'	94'	110,000 lb
737-200	130	93'	100'2"	115,500 lb
737-300	135	94'9"	109'7"	139,500 lb
737-400	139	94'9"	119'7"	150,000 lb
737-500	127	94'9"	101'9"	136,000 lb
737-600	126	112'7"	102'6"	144,500 lb
737-700	132	112'7"	110'4"	133,000 lb
737-800	142	112'7"	129'6"	155,500 lb
737-900	141	112'7"	138'2"	164,000 lb

Due to the low number of different aircraft types that have been identified in the dataset, the extent of low data quality in the aircraft dataset as described in Section 3.2.2 is drastically reduced. Especially for the few aircraft models located in the dataset, missing values are easily filled in manually by looking up data sheets for corresponding models. However, the scraped data did not include the sub-model number for retrieved aircraft models regarding their tail numbers. Since this information is missing, actual aircraft characteristics of a given tail number is not available for the training process. New versions of popular aircraft models are released regularly, which results in a significant variance between versions of one aircraft model. These deviations are depicted in Table 3.6 and shows each 737 submodel's approach speed, wingspan, length and maximum take-off weight without winglet configuration. The dimensional attributes are retrieved from spec sheets provided by Boeing [37] and approach speeds were taken from the FAA reference code [38]. On Average, an individual aircraft is capable of staying in operation for 25 years, which overlaps with the lifecycles of aircraft from newer models [39]. Therefore, various versions of the *Boeing 737* for example are encountered in the field simultaneously. To guess the characteristics of each aircraft regardless of their submodel, only the most recent versions' dimensional features with good data qualities were chosen and inserted for each aircraft model. Due to unknown actual dimensions for the aircraft in the dataset, aircraft dimensions are relevant for perturbation option definition as stated in Chapter 4.

3.3.3 Preprocessing METAR Data

The downloaded METAR data for the ATL airport includes information about specific weather events around the airport, which provides useful information for the prediction of aircraft delays [22]. The dataset's quality is good, with little unexplainable *null* values. However, starting in May 2016, wind and pressure sensors are no longer delivered in an hourly aggregated form, instead being updated in 5 minute intervals. The difference in sensor report frequency from wind and

pressure sensors compared to other sensors had to be acknowledged. To match the hourly structure, all entries are aggregated onto the full hour in which the measurement occurs. For example, a measurement being reported at 12:35 will be aggregated with all other measurements within the 12th hour. For all numerical values, median was chosen as aggregation method due to its property of being more outlier resistant. This being especially important for volatile features such as wind direction and wind speed. For categorical values, mode was used, to capture the most prevalent weather situation within the hour.

Information about weather events is provided within the METAR string, as seen in Table 3.3. To obtain information about the occurrence of an event within a METAR string, each event code listed in Table 3.4 is found by matching the code within the METAR strings in addition to their strength indicators that are added as prefixes. Since the occurrence of events is not mutually exclusive within each hour, such weather events need to be handled separately to the other categorical features and cannot simply be aggregated by mode operations. As described in Section 3.3.3, METAR events have an additional indicator for event strength at the time of measurement. To take these into account, all extracted event codes and indicators were one-hot encoded such that for each hourly interval, '1' is inserted whenever the combination of event code and indicator occurs at least once. This one-hot encoding adds $16 \cdot 3$ columns to the dataset for each event code extracted with each intensity indicator or lack thereof. To minimize the number of columns needed to represent weather events and yet still allow for a differentiation between, for example, '-RA' (weak rain) and '+RA' (strong rain), the one-hot encoded values were multiplied by their respective indicator values where the indicator values are exchanged with numerals from 1 to 3 with 3 indicating the strongest possible event and 1 the weakest. This multiplication with intensity indicators leads to 16 data columns representing weather event occurrence within an hour of a METAR report. '0' indicates the absence of an event, while '1', '2' and '3' indicate that at some point within the hour a corresponding event was measured and rated as weak, moderate or heavy respectively. On the occurrence of equal weather events with varying intensities within an hourly window, the assumption that strong weather events impact flight scenarios stronger than weaker events is taken. Therefore, the max function is used for aggregation, only keeping the strongest possible events within the dataset.

3.3.4 Preprocessing NOTAM Data

The dataset formed by NOTAM reports inherits the same characteristics as previous datasets when it comes to datetime features. Date features are stored as strings in the form of 'MM/DD/YYYY hhmm'. The time of day and date strings need to be separated for further processing for all relevant date features, namely *'Effective Date'* and *'Expiration Date'*. The feature *'Issue Date'* is discarded from the dataset since it does not provide information about scenarios affecting air travel at a given time, rather only documenting the time of transmission of a report. For messages with immediate effect, the issue date is equal to the effective date. For this reason, the effective date is sufficient for model training purposes.

To make NOTAM information readable for the machine learning algorithm without using Natural Language Processing, the semi-structured text outputs of the NOTAM reports need to be converted into a numerical format. NOTAMs cover a multitude of message types, ranging from alerts about malfunctioning equipment to informing terminal personnel and aircraft crew about

altered communication frequencies, and many more scenarios. Since this thesis focuses on an individual flight arrival delay prediction, only reports indicating events which could influence arrival delay are relevant for this task. By experimentally looking at the various categories of NOTAM reports, it was chosen to select only NOTAM messages that mentioned specific runways. By filtering messages to include only the keywords *'CLSD'*, *'FICON'* and *'OUT OF SERVICE'* it is possible to generate a list of NOTAMs covering runway closures, bad conditions of the runway and malfunctioning runway equipment respectively. Furthermore, these keywords are mutually exclusive, allowing us to map an indicated runway to the respective keyword within the timeframe from NOTAM effective date to expiration date. To systematically capture the operational status of the five runways at Hartsfield-Jackson Airport, a $5 \cdot N$ matrix was constructed, where N represents the number of NOTAM entries in the dataset and 5 being the number of different runways present at ATL in 2017. Therefore, each row of the matrix corresponds to a specific runway, indicated by identifying the forementioned runway code in the message, with entries inserted based on the occurrence of relevant keywords according to the following encoding scheme: *CLSD* $\rightarrow 3$, *OUT OF SERVICE* $\rightarrow 2$, and *FICON* $\rightarrow 1$. This structure transforms NOTAM data from textual inputs into numerical values. These numeric values are mentioned as *'obstacle indicators'* in subsequent sections and are interpretable by machine learning algorithms. These indicators represent information held within NOTAMs coarsely, which would be outperformed by more sophisticated methods. However, since this thesis focuses on the perturbation analysis, model performance is of secondary importance, which makes this simplistic approach to NOTAM analysis sufficient for the task at hand.

3.3.5 Data Integration

To generate a data source which can later be split into training and test data, the previous established data sources need to be integrated into a combined data table. This chapter documents the process of data integration by explaining the logic by which data has been combined, while also showing challenges and information that was lost due to the used process. As shown in previous sections, the dataset consists of (i) flight data, (ii) aircraft data, (iii) METAR data and (iv) NOTAM data. Since flight data forms the base of the classification task, the other datasets are merged into it one after the other.

Aircraft Data Integration

Since the flight dataset contains tail numbers to identify carriers, these tail numbers were chosen to find the corresponding aircraft models. This process allows us to merge flight data with aircraft data by model name after identifying the models for each tail number. All tail numbers which were not included in the aircraft data were discarded. Out of all 739.429 rows within the flight dataset, 16.864 tail numbers could not be resolved to an aircraft model, which equals to a reduction of size regarding flight data entries by 2.3%.

METAR Data Integration

The METAR data set contains weather data aggregated for every hourly interval. The flight data set includes precise timestamps for the planned arrival time at the destination airport, where corresponding weather data is recorded. Therefore, each flight entry is matched to the nearest

hourly weather record based on its planned arrival time within the METAR dataset. This approach is sufficient since previous preprocessing steps ensured that documented weather conditions remain relatively stable within an hour due to the aggregation processes defined in previous sections.

NOTAM Data Integration

NOTAM data, which holds information about runway obstruction levels for each runway, includes an effective date and an expiration date. Within this date range, a given scenario may occur for a specific runway. To combine NOTAM data with flight data, every flight data entry was mapped to all NOTAM entries which were effective during the planned departure of the flight, following the logic of *'NOTAM_Effective_Date'* $\leq$ *'Flight_Departure_Date'* $<$ *'NOTAM_Expiration_Date'*. This mapping created a $n_{flight} \cdot n_{notam}$ boolean matrix, where n_{flight} equals the number of entries within the flight data and n_{notam} equals the number of entries within the NOTAM data set. Each element in the matrix is either 1 when the given flight falls within the effective date range of the corresponding NOTAM entry, or 0 when it does not. This matrix allows fast identification of corresponding NOTAM events to each flight. By iterating over the boolean matrix, all 'obstruction values' explained in the NOTAM preprocessing section for each NOTAM are inserted for each flight. Lastly, all identified 'obstruction values' mapped to each flight were aggregated to allow single row presentation of the information. Since the most critical obstruction has the supposed strongest influence on a given flight, similarly to the treatment of weather events, the max function was used as an aggregation method. Finally, each flight now contains information about the extent of runway obstruction as described by the processed NOTAM data at the time of departure. After these processes, the data set is combined into a single table with 348.140 rows and 47 columns, as depicted in Table 3.7 which shows all columns with their respective ranges and example values for illustration. Within the table, the ordinal features holding the weather events and runway obstacle indicators are grouped in a single row to allow for a more compact table representation.

3.3.6 Preprocessing on Combined Dataset

As previous sections documented the creation of a single data table out of various data sources, this section explains additional preprocessing steps taken on the merged data table to further improve data quality of the training-ready input data set and perform operations previously not possible due to the missing links between the datasets.

Cyclic Time-Series Representation

One of the most important features for flights are departure and arrival times of carriers. This thesis heavily relied on time information to combine datasets, making time information essential for this carrier delay prediction task. Therefore, allowing machine learning algorithms to correctly interpret time-sensitive data is critical. A given flight for a classification scenario might operate over midnight, which would skew distances between data points on categorical representations of date objects. For example, a flight arriving on Sunday 11pm is more distant to a flight arriving on Monday than a flight arriving on Wednesday in numerical representation. This problem exists for both minute-of-day and day-of-week features of our data. However, other works have shown that time of day and day of week are capable predictors for delay classification. To better represent distances between time-sensitive data points, time specific data was encoded cyclical in sine and

Table 3.7: Features and data ranges for the combined dataset.

Name	Range	Example
OP_UNIQUE_CARRIER	[-11;-3]	-11
ORIGIN	[-19;2]	-8
DEP_DELAY	[-188;1816]	-2
CRS_ELAPSED_TIME	[45;530]	223
DISTANCE	[83;4502]	1590
CRS_DEP_DATE	[2017-01-01;2017-12-31]	2017-01-01
CRS_ARR_DATE	[2017-01-01;2018-01-01]	2017-01-01
ACT_DEP_DATE	[2017-01-01;2018-01-01]	2017-01-01
ACT_ARR_DATE	[2017-01-01;2018-01-01]	2017-01-01
PREV_FLIGHTS_1H	[0;120]	4
ARR_DELAY	[-202;1810]	-7
Physical Class	[Jet;Turboprop]	Jet
Engines	[2;4]	2
Approach Speed	[114;157]	141
Wingtip Config	[no winglets;wingtip fences]	no winglets
Wingspan (ft)	[65.75;199.92]	112.58
Length (ft)	[86.42;242.33]	138.17
Tail Height (ft)	[20.75;64.25]	41.42
MTOW	[41888;660000]	174200
Parking Area (sf)	[5682;48446]	15555
tmpf	[15.1;93.9]	42.1
dwpf	[5.0;75.9]	36.0
relh	[12.7;100.0]	78.8
drct	[0.0;360.0]	150.0
sknt	[0.0;33.5]	6.0
p01i	[0.0;0.915]	0.01
alti	[29.16;30.7]	30.11
mslp	[987.1;1041.2]	1020.1
vsby	[0.13;10.0]	10.0
feel	[1.11;99.1]	37.69
EVENT_BR - EVENT_TS	[0;3]	0
08L/26R - 10/28	[0.0;3.0]	1.0

cosine parts for both week-of-month and minute-of-day representations. This enables proper representation of distances between data points which are going over the 24th hour of the day or the seventh day of the week with the disadvantage of increasing the dimensionality of the dataset [40]. Figures 3.4 and 3.5 show all possible data points in the sine-cosine-transformed space for the arrival time features, which were calculated based on the equation as follows:

$$x_{sin} = \frac{\sin(2 \cdot \pi \cdot x)}{v_{max}}; \quad x_{cos} = \frac{\cos(2 \cdot \pi \cdot x)}{v_{max}}; \tag{3.1}$$

where x represents a given input for either day-of-week or minute-of-day and v_{max} represents the number of possible values x could represent. 1440 for minute-of-day and 7 for day-of-week.

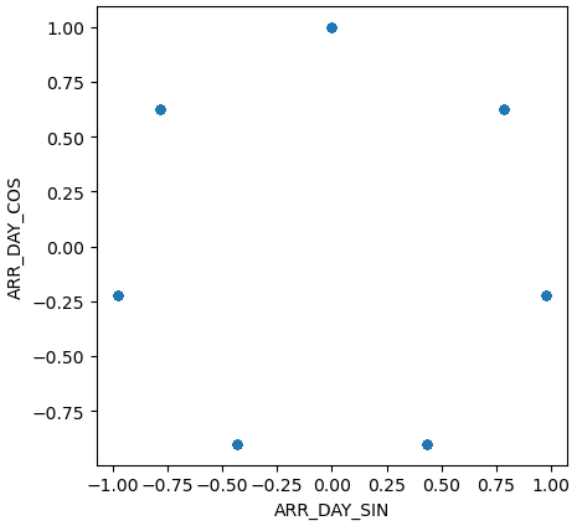


Figure 3.4: Sine-Cosine transformed representation of arrival day of week.

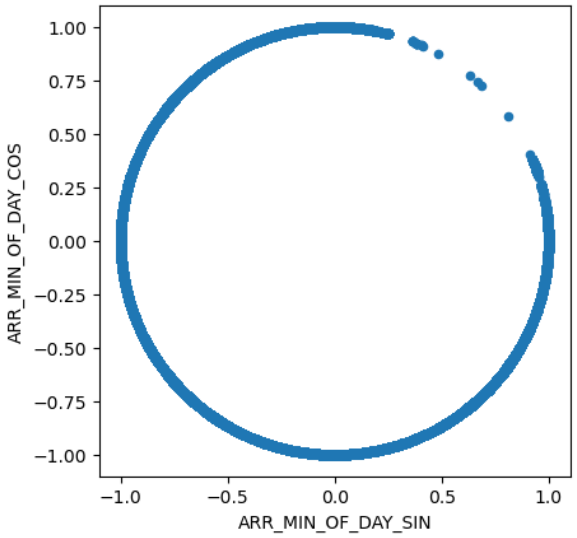


Figure 3.5: Sine-Cosine transformed representation of arrival minute of day.

It is important to note that the time information depicted in Figures 3.4 and 3.5 shows the planned arrival time which remains to be the only time relevant feature later used for training since actual arrival date is unknown at the time of departure, the diversion between actual and planned departure is represented by a much simpler feature in *'departure delay'* and in conflict whether to choose planned departure or planned arrival, the latter performed slightly better in experimental training runs.

Renaming of Feature Names

Due to the incorporation of multiple data sources, namespaces and units of data, features are not uniformly named and given in SI units. For better clarity and data understanding, the names of

features were simplified and equipped with a unit indicator such that all features are represented in the form following *'FEATURE_NAME(UNIT)'*. Furthermore, all units that captured sensor data which were not held in metric units were changed into metric units to improve readability and comparability.

Variance-Based Feature Reduction

Due to the phenomenon called curse of dimensionality, it is not beneficial to a model's performance to keep features which are adding less to the model's performance than their addition costs [41]. It is oftentimes difficult to quantify this cost-benefit-ratio, but one can assume that especially features with a low variance are more prone to have negligible impact on performance while costing additional feature space. Figure 3.6 shows the corresponding feature variances within the training dataset after min-max scaling. At first, it was chosen that all features falling under an experimental threshold, as shown by the horizontal blue line at 0.01, should be discarded from the dataset for training purposes. This procedure was far too limiting and resulted in the deletion of some majorly important features such as *'DEP_DELAY(MINS)'* which's importance is shown later. Nonetheless, the low-variance features *'NR_ENGINES'* and *'JET(YN)'* did not provide enough information for them to be useful for later analysis. Although having a lower variance value-wise, low-variance event-features were kept within the data because of the assumption that weather events are inherently rare, but might impact flights strongly whenever they occur.

Correlation-Based Feature Reduction

Similar to variance-based feature reduction, correlation can be used to eliminate features that are represented better within the dataset by related features. Figure 3.7 shows the correlation matrix of the dataset at this stage of the preprocessing process. Some notorious clusters are the carrier specification features, atmospheric temperature and humidity measurements and some event

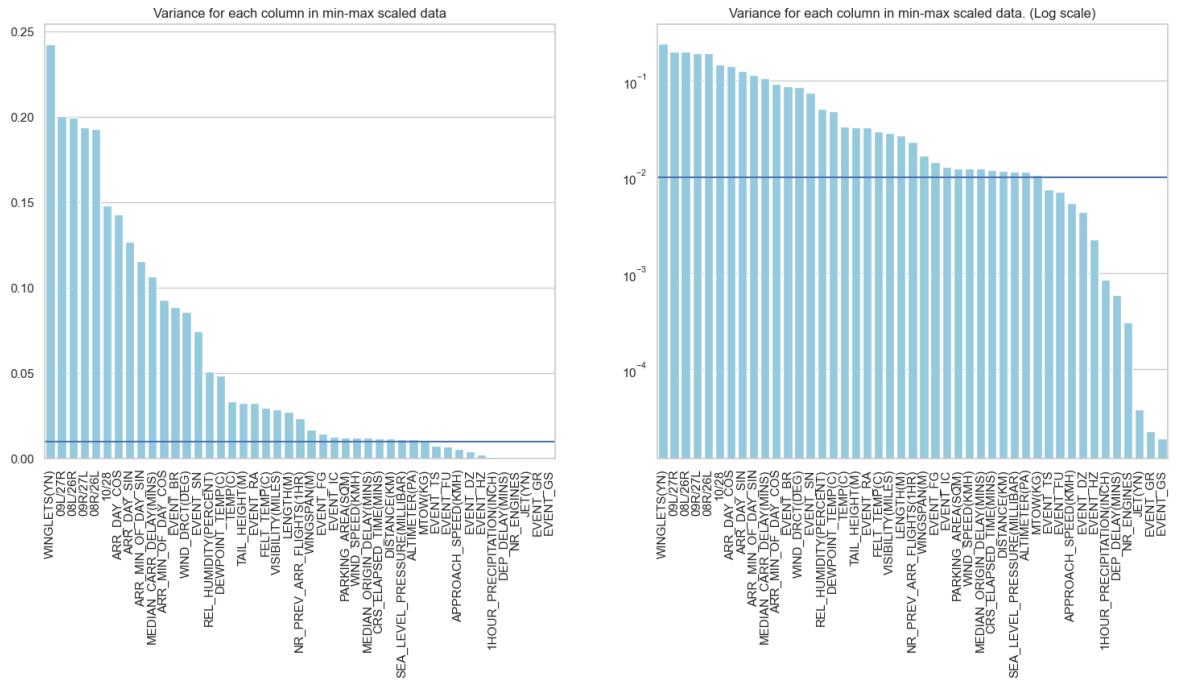


Figure 3.6: Variances of features in absolute (left) and logarithmic (right) scale.

features that are correlating to each other. Importantly, *'DEP_DELAY(MINS)'* is the only feature correlating with the target variable, which further fortifies the previous assumption that it should be kept within the dataset despite its low feature variance. For the carrier specifications, it was chosen to only keep *'PARKING_AREA(SQM)'* and *'TAIL_HEIGHT'*. The earlier represents a multiplicative representation of *'WINGSPAN(M)'* and *'LENGTH(M)'*, making these features obsolete, while the latter is the only carrier specification feature which does not correlate almost perfectly with *'PARKING_AREA(SQM)'*. Within the temperature measurement cluster, *'FELT_TEMP(C)'* is discarded since it correlates perfectly with *'TEMP(C)'*, which according to the ASOS User Guide is the more objective and precise measurement between these two [29]. Lastly, the two atmospheric pressure measurements *'ALTIMETER(PA)'* and *'SEA_LEVEL_PRESSURE(MILLIBAR)'* as well as the two route distance measures *'DISTANCE(KM)'* and *'CRS_ELAPSED_TIME'* correlate strongly. For the former, *'SEA_LEVEL_PRESSURE(MILLIBAR)'* was chosen due to better interpretability of the value and better data quality within the feature. For distance measures, the decision to drop either of the features was delayed to the importance-based feature reduction. Both features correlate strongly, which eventually leads to one of the features to be dropped. However, because one feature is represented by time and the other by bee-line distance, they show different scopes. Dropping one of these features would be arbitrary when solely relying on the correlation matrix.

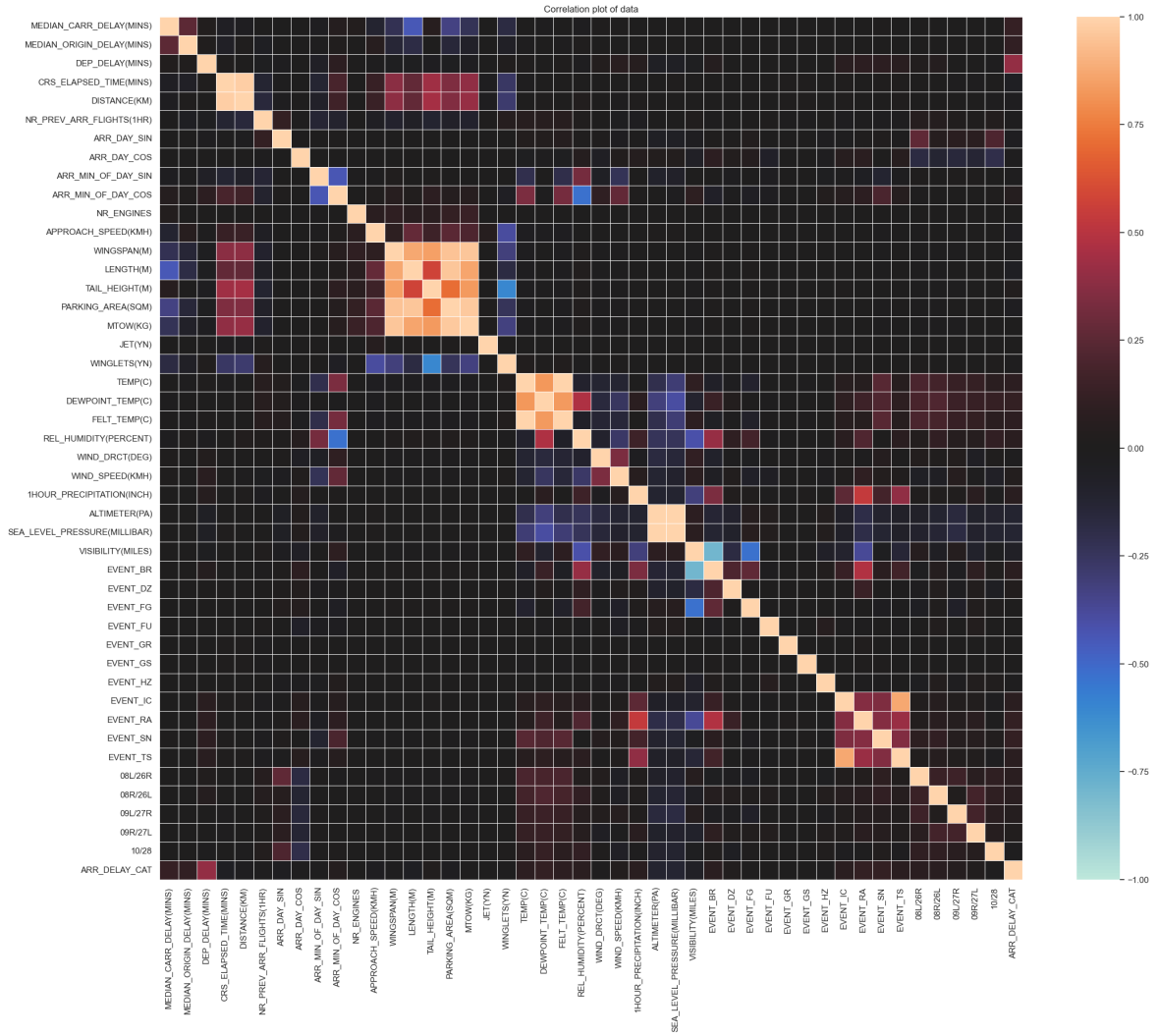


Figure 3.7: Correlation matrix of input data with target variable.

Importance-Based Feature Reduction

Initially, simple feature reduction methods were used to find low performing features, reducing the curse of dimensionality. Correlation analysis and variance filtering were used as described in the previous sections. After the first models were evaluated, insufficient model performances led to a step back into the data preprocessing phase to reevaluate dimensionality reduction techniques. Through further iterations of trying different parameters, the best performance was achieved by using an importance-based feature reduction technique using a primitive random forest regressor. Although this thesis focuses on classification, the data at hand includes carrier arrival delay in minutes, also allowing for regression tasks. For feature reduction, regression was chosen since differences between features were more easily identified by the random forest regressor than it was for the classifier. Both the random forest regressor and cross validation algorithms used for this task were taken from the sklearn python library. All used parameters for this process are shown in Table 3.8. Parameters not mentioned in Table 3.8 were not altered from the default sklearn implementation. The max_depth feature was selected due to the results of the elbow test seen in Figure 3.8. This test shows that after max_depth $= 4$, only marginal gains in performance are observable, while time needed for each run increased steadily. The parameter max_features $= 1.0$ was chosen to investigate the whole feature space for each run. These simple parameters were then selected to allow a decent score while still maintaining manageable training times.

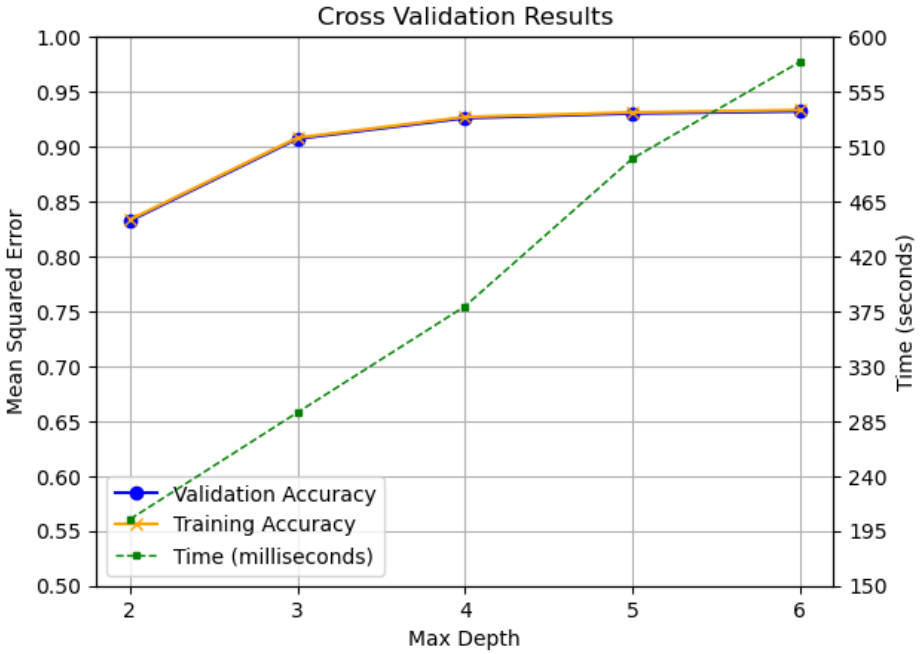


Figure 3.8: Elbow test to find optimal max depth parameter for feature reduction regressor.

Random Forest		Cross Validation	
Param Name	Param Value	Param Name	Param Value
n_estimators	100	scoring	neg_root_mean_squared_error
max_features	1.0	cv	5
max_depth	4		

Table 3.8: Parameters of regressor and cross validation for feature reduction.

The process of feature elimination was performed by forming the mean of calculated root mean squared errors (RMSE) of 5-fold cross validation over the full feature space. Then, for each iteration, a feature is removed to calculate the RMSE that is achieved without the corresponding feature. This process repeats for each feature in the training set, showing each potential gain or loss in respect to the absence of the respective feature. Figure 3.9 demonstrates the change in RMSE of the random forest regressor due to the removal of each feature, highlighting their relative importance to the model's predictive performance. Values above zero indicate that the absence of the feature led to higher RMSE, while values beneath zero indicate an improvement after the respective feature was dropped.

As demonstrated in Figure 3.9, almost all features negatively impacted the regressor's RMSE for the first iteration of the feature elimination process. By starting to eliminate the worst performing feature and repeating the previous process for the remaining feature space, Figures 3.10 and 3.11 show, that those features which are initially negatively impacting the model's performance, are influencing the achievable RMSE positively in future iterations with the slightly smaller feature space. This iterative feature elimination process is performed until all features show a positive influence to the RMSE, resulting in the final feature space shown in Figure 3.11. Noteworthy, throughout all iterations, the feature *'DEP_DELAY(MINS)'* is shown to be the most important feature for the model's performance. The high importance of the departure delay feature is likely due to its stronger correlation with the target variable compared to other features.

According to the described process of feature elimination, the two worst performing features in the dataset were the created median delays, namely *'MEDIAN_ORIGIN_DELAY(MINS)'* and *'MEDIAN_CARRIER_DELAY(MINS)'*, which were based on data of the previous year. All other features positively impacted the chosen regressor in the final iteration, which according to this process concludes the feature elimination step. Furthermore, the findings of this process

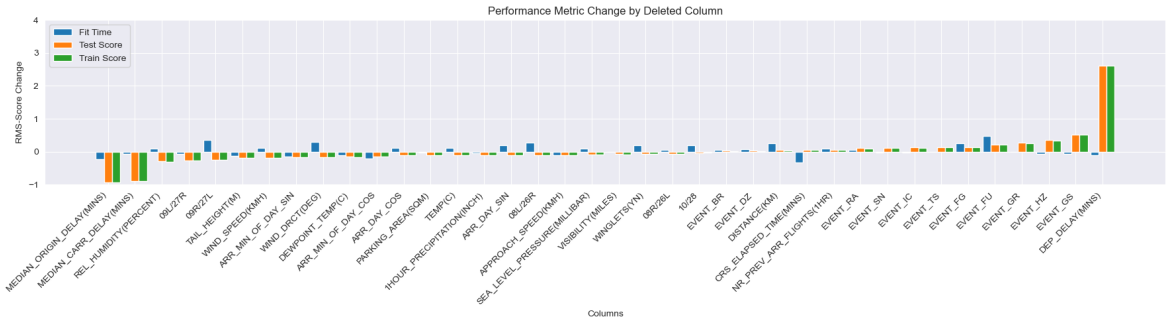


Figure 3.9: First iteration of feature reduction process.

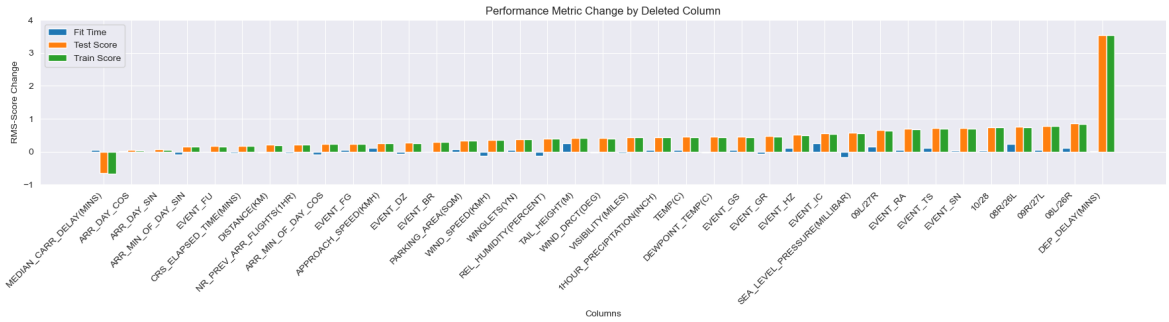


Figure 3.10: Second iteration of feature reduction process.

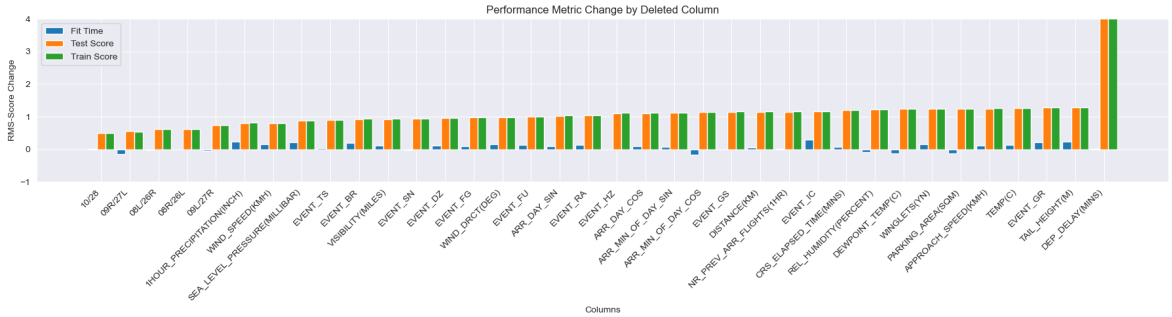


Figure 3.11: Third iteration of feature reduction process.

identified the feature *'DISTANCE(KM)'* to be slightly less important to the regression task than *'CRS_ELAPSED_TIME(MINS)'*. Therefore, the former is dropped from the dataset, making *'CRS_ELAPSED_TIME(MINS)'* the only remaining distance attribute in the dataset.

Feature Combination of NOTAM Columns

Although the process of feature elimination did show that no single remaining column negatively impacts the chosen regressor, this does not guarantee the best possible input matrix for the model training in later stages. Specific combinations of features might improve model performance further. Since the NOTAM features have the least impact on the regressor as shown in Figure 3.11 looking for an improvement for this representation was the highest priority. NOTAM features consist of an obstruction indicator that is assigned to each runway at the time of departure. By eliminating the indicator for each runway and instead forming a percentage of affected runways, the number of features needed to represent NOTAM information is reduced. With that, an input vector over the five possible runways at ATL would be transformed from [0,2,1,0,1] to the single value [0.6] since 60% of the runways have some instance of impediment at the time of departure within the NOTAM reports in this example. By collapsing the runway obstruction indicators into a single column, the information about the severity of the obstruction is lost. However, due to this feature transformation, a slight improvement for RMSE was achievable, as shown in Figure 3.12 for both training and test data after 5-fold-cross-validation. Additionally, fit times decreased slightly, which further increases the benefit of transforming NOTAM data into the combined representation.

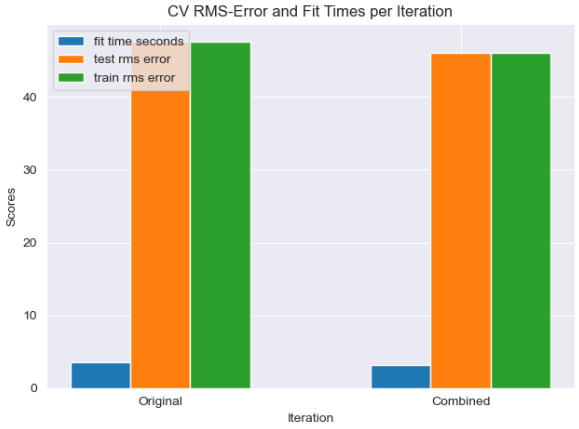


Figure 3.12: RMSE of combined NOTAM set (right) vs. runway specific NOTAM set (left).

Target Variable Creation

In previous preprocessing steps, a regressor was used to find patterns and reduce the dimensionality of the dataset. For the next steps of model training, the cardinal target variable depicting delays in minutes is broken down into delay categories according to the FAA guidelines which identify a flight arriving 15 minutes before the planned arrival as early arrival and a flight arriving 15 minutes after the planned arrival as late arrival. Therefore, going forward, the target variable is formulated based upon the logic depicted in Equation 3.2.

$$y \mapsto \begin{cases} \text{early}, & \text{if } y \leq -15, \\ \text{on\_time}, & \text{if } -15 < y < 15, \\ \text{late}, & \text{if } y \geq 15. \end{cases} \tag{3.2}$$

By following this logic, the target variable distribution shown in Figure 3.13 is transformed into the binned distribution shown in 3.14. As evident in Figures 3.13 and 3.14, the classification task introduces an imbalance within the target class since most flights arrive punctual, which is usually accounted for by sampling methods such as oversampling or undersampling. However, sampling techniques did not improve evaluation metrics by a wide enough margin to justify the data loss in undersampling or the higher complexity and training times that come from oversampling or generative approaches to sampling. Therefore, the data imbalance was kept and acknowledged in the modelling phase of the CRISP-DM process.

Final Data Set

After all previously explained data gathering, transformation and preprocessing steps, the final dataset is represented by a $329.005 \cdot 31$ data table with attributes as displayed in Table 3.9. Due to the weather sensor and time data attributes, most attributes have a cardinal scale with only one categorical scale feature with the binary field *'WINGLETS(YN)'* and the ordinal features comprised of the weather event indicators and the target variable for classification. Noteworthy, almost all ordinal features indicating weather events except for *'EVENT_TS'* and *'EVENT_RA'* do not include a single data entry showing a strong indication of a given event to happen, resulting in a max value of '2' instead of the theoretically possible '3'. This shows that extreme weather

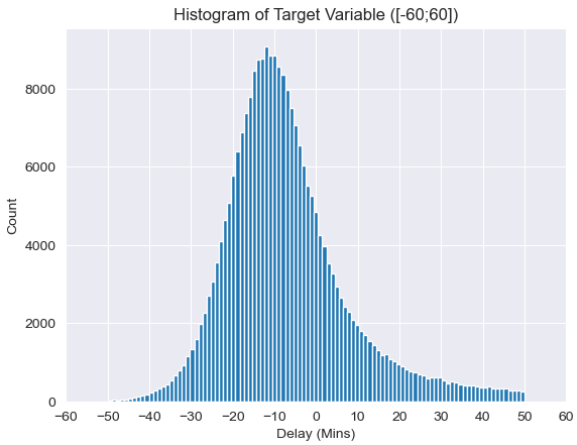


Figure 3.13: Distribution of cardinal target variable.

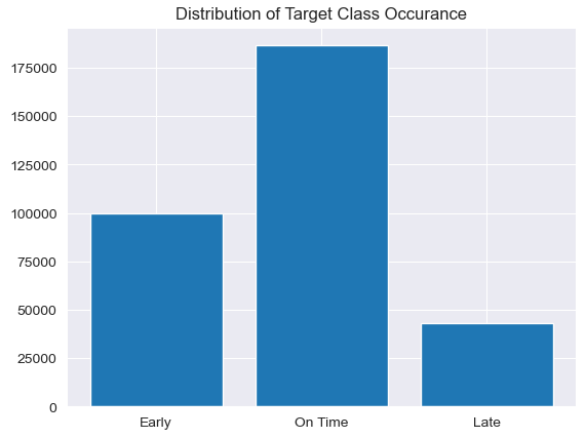


Figure 3.14: Distribution of binned target variable.

scenarios are rare, despite using $max(x)$ to aggregate overlapping scenarios, which increases the proportional occurrence of higher intensity indicators in the dataset. Lastly, the dataset was cleaned in previous steps such that no data entry is empty or is represented by strings. Because of this, the dataset is usable by most machine learning algorithms, allowing easier experimental settings for trying out various models. Although, some models require scaling, outlier detection and handling of the data imbalance, the clean dataset is an important first step to allow for a streamlined model building process.

3.4 Modelling

Previous sections explained steps required to obtain a clean dataset ready for the model training process. This section describes the tasks taken during the modelling phase of CRISP-DM. The trained machine learning models, coming out of this CRISP-DM phase, are being evaluated and deployed in subsequent phases of the data mining process. First, an overview of the machine learning algorithms used within the modelling phase of the CRISP-DM process is being established. Although multiple simple classifiers have been involved in the full scope of this thesis, only the final selection of models trained on the previously described feature combination is explained. Following this, the selection process of algorithms is briefly explained. Next, the process of parameter fine-tuning and the final parameters are shown. Finally, the training process that resulted in the creation of the final models is described. The performance scores regarding the model's capability of predicting flight delays on the test dataset is explained in the following evaluation section. All models that are being explained in this section are ultimately used to perform reliability assessment of individual predictions by the perturbation approach, as explained in Chapter 4. Therefore, the models are the basis for the perturbation approach, which is the focal point of this thesis.

3.4.1 Overview of Algorithms

For the scope of this thesis, it was decided to use machine learning algorithms provided by the sklearn python library. At early stages of the thesis, ensemble and boosting methods were not evaluated, which resulted in the limitation to use rather simple algorithms like *Tree Classifiers*, *Support Vector Classifiers* and *Naive Bayes Classifiers*. Since the previously mentioned classifiers have fundamentally different mechanics, providing input data that works equally well for all

Table 3.9: Attribute properties of preprocessed dataset.

Name	Scale	Range	Example
DEP_DELAY(MINS)	cardinal	[-188;1816]	107
CRS_ELAPSED_TIME(MINS)	cardinal	[45;530]	89
NR_PREV_ARR_FLIGHTS(1HR)	cardinal	[0;120]	74
ARR_DAY_SIN	cardinal	[-0.975;0.975]	-0.434
ARR_DAY_COS	cardinal	[-0.901;1.0]	-0.901
ARR_MIN_OF_DAY_SIN	cardinal	[-1.0;1.0]	-0.986
ARR_MIN_OF_DAY_COS	cardinal	[-1.0;1.0]	0.169
APPROACH_SPEED(KMH)	cardinal	[211.128;290.764]	255.576
TAIL_HEIGHT(M)	cardinal	[6.325;19.583]	9.001
PARKING_AREA(SQM)	cardinal	[527.871;4500.743]	1525.177
WINGLETS(YN)	categorical	[0;1]	1
TEMP(C)	cardinal	[-9.389;34.389]	6.722
DEWPOINT_TEMP(C)	cardinal	[-15.0;24.389]	-3.278
REL_HUMIDITY(PERCENT)	cardinal	[12.7;100.0]	48.88
WIND_DRCT(DEG)	cardinal	[0.0;360.0]	290.0
WIND_SPEED(KMH)	cardinal	[0.0;62.042]	26.854
1HOUR_PRECIPITATION(INCH)	cardinal	[0.0;0.915]	0.0
SEA_LEVEL_PRESSURE(MILLIBAR)	cardinal	[987.1;1041.2]	1018.8
VISIBILITY(MILES)	cardinal	[0.13;10.0]	10.0
EVENT_BR	ordinal	[0;2]	0
EVENT_DZ	ordinal	[0;2]	0
EVENT_FG	ordinal	[0;2]	0
EVENT_FU	ordinal	[0;2]	0
EVENT_GR	ordinal	[0;2]	0
EVENT_GS	ordinal	[0;2]	0
EVENT_HZ	ordinal	[0;2]	0
EVENT_IC	ordinal	[0;2]	0
EVENT_RA	ordinal	[0;3]	0
EVENT_SN	ordinal	[0;2]	0
EVENT_TS	ordinal	[0;3]	0
RUNWAY_ERROR(PERC)	cardinal	[0.0;1.0]	0.4
ARR_DELAY_MINS	ordinal	[0;2]	2

classifiers turned out to be challenging. Particularly, feature combinations, sampling methods and data scaling operations added variability between the trained models' performances, depending on how the corresponding model responded to the different operation's outcome. After some experimental runs, it was evident that tree algorithms seemed to work best with the prepared

dataset. Additionally, the tree algorithms did not require sampling and scaling methods for good performance, leading to less complexity in the training pipeline. To further improve the model performance, boosting and bagging methods showed the highest potential, reducing the model types to the three tree-based algorithms namely, *Random Forest Classifier* (RF), *Adaptive Boosting Classifier* (ADAB) and *Gradient Boosting Classifier* (GB). Since the focus of this thesis lies heavily on the demonstration of the perturbation approach, it was decided to stick with these three tree-based models without further evaluating more complex models. Furthermore, the classifier's respective performances showed to already beat the target baselines. Therefore, the modelling phase was concluded without evaluating further classifier types. Since the selected classifiers rely on fitting parameters to enable their full potential on a particular training set, parameter fine-tuning was conducted. For speedy initial parameter fine-tuning, *Halving Grid Search* [42] was initially chosen. Finally, *Exhaustive Grid Search* was used to narrow down optimal classifiers by checking parameters around initially estimated parameter values. These parameters were then seen to be the best possible parameters for a given estimator dataset pair. The following subsections explain each used algorithm's functionality broadly, while also showing benefits and shortcomings for each method.

Random Forest Classifier

The idea of random forests was initially proposed by Breiman [43], who extended the idea of forest ensemble methods by training a number of fully grown tree classifiers on a random subset of the training data. By relying on the results of multiple decision trees, a more accurate and stable result is achievable compared to single tree classifiers. Random forests are capable of classification and regression tasks. For classification, the final prediction is determined by majority voting across all decision trees within the forest, whereas for regression, the average of all predicted values is used as the final output. This voting or averaging combined with random sampling and random feature selection for each decision tree reduces the variance of the prediction, making the random forest less prone to overfitting and more robust to noise in the dataset. Additionally, the classifier inherits the decision tree's characteristics of being able to classify categorical data, imbalanced data and data with missing values equally well [44]. This, combined with the reduced risk of overfitting, makes random forest classifiers easy-to-use and robust classifiers. They are capable of reaching good accuracy on a wide variety of datasets, not needing particular preprocessing steps to achieve decent results.

Random Forest Classifier, as provided by the sklearn python library, is a highly adaptable classifier with 19 possible parameters to alter the classifier's behaviour. However, it has been identified, that the most influential parameters are (i) *n_estimators*, (ii) *max_depth*, (iii) *max_samples*, (iv) *max_features* and (v) *class_weight*. The first parameter *n_estimators* influences the number of trees that are built within the forest. With a higher number of estimators, the achieved accuracy increased up to a certain point, after which the performance of the random forest stopped to increase significantly while still gaining on training time. For the parameter *max_depth*, it is critical to choose a depth that is deep enough to find complex patterns in the training data, while also being shallow enough to prevent overfitting on training data patterns. The parameters *max_samples* and *max_features* are controlling the maximum samples and features that are considered for splitting individual decision trees. Lower numbers for each of these parameters increase the randomness of the forest, lowering overfitting tendencies. Lastly, the parameter *class_weight* can be set to add weights to the classes in order to alleviate class imbalances. This addition is not available for the other ensemble methods described in this section and is partially responsible for the *Random Forest Classifier*'s decent accuracy on the minority classes as shown in Section 3.5.

Adaptive Boosting Classifier

Adaptive boosting was proposed by Freund and Shapire [45] and is an ensemble method for classification and regression problems consisting of decision trees which is similar to random forest classifiers. Contrary to random forest classifiers, the decision trees that are built for adaptive boosting are weak learners themselves, since they usually only consist out of the base and leafs. Because of their shallow depth, such decision trees are also called decision stumps. Furthermore, the decision stumps are trained iteratively such that each subsequent stump learns from misclassifications of their predecessor. For each iteration, weights on each of the training data observations are updated to better fit previously misclassified observations [46][47]. For the final prediction of the whole ensemble, the vote is weighted such that more accurate decision stumps have more say in the prediction. Similarly to random forests, the ensemble method reduces the tendency to overfitting and works with binary, categorical and nominal data values. Although adaptive boosting is generally considered to be resistant to overfitting, compared to other boosting and bagging methods, it has a tendency to fixate on outliers due to focusing on training observations that are harder to fit onto [48]. This fixation on outliers allows adaptive boosting to be usable for outlier detection, but introduces challenges for the task within this thesis. Additionally, adaptive boosting requires some sort of base classifier from which the process starts. The complexity and type of the base classifier influences the generalizability of the resulting classifier strongly.

The sklearn implementation of the *AdaBoost Classifier* includes three parameters that alter the classifier's behaviour, namely (i) *estimator*, (ii) *n_estimators* and (iii) *learning_rate*. The first parameter *estimator* defines the base classifier from which the ensemble starts to build off on. If none is selected, a base *Decision Tree Classifier* with *max_depth* $= 1$ is chosen. This base classifier was not altered from the default for the final implementation of the *AdaBoost Classifier* used in this thesis. The second parameter *n_estimators* limits the number of classifiers created within the ensemble. Lastly, *learning_rate* determines the speed at which the weights are changed

for each boosting iteration. The parameter is given as float within the range of $(0.0, \infty)$. With a higher *learning_rate* each individual classifier gains increasing contribution to the final prediction.

Gradient Boosting Classifier

Gradient boosting was proposed by Friedman [49] who used the boosting approach on decision trees with a different training procedure than it is the case for the previously explained adaptive boosting approach. In the first step of the boosting process, an initial prediction is constructed by minimising a loss function over all training entries [47]. Then, residuals based on the true target labels and the initial prediction are constructed. After calculating the initial set of residuals, an iterative process starts, within which n weak tree regressors are sequentially constructed. Each of these regressors tries to fit the training data onto the calculated residuals of their predecessor, steadily correcting the respective output. After all n weak learners are constructed, the training process concludes. When a new data entry is to be predicted by the ensemble, the initial prediction is added with all outputs of each iteratively constructed tree, producing a final output for a given data entry. For classification, the output is transformed into the corresponding target class. Gradient boosting is known to perform well after careful hyperparameter tuning. However, gradient boosting is also prone to overfitting compared to other boosting methods and can be computationally expensive [47]. The overfitting problem is alleviated by adding a learning rate η to each of the weak tree regressor's output. For the classification task of this thesis, gradient boosting showed to perform better than the other evaluated methods, even though the fixating on the majority class is observable, as seen in future sections.

The *Gradient Boosting Classifier*, as provided by sklearn, has 20 potential parameters. The three parameters (i) *n_estimators*, (ii) *learning_rate* and (iii) *max_depth* showed to have the biggest influence on the predictive performance of the model. The parameter *learning_rate* is given as a float within the range of $[0.0, \infty)$ and adds weights to the outputs of the correcting estimators. The parameter *max_depth* and *n_estimators* are equivalent to the parameters of the *Random Forest Classifier*.

3.4.2 Modelling Process

Finding the correct model and parameters for a given machine learning project can turn into an exhaustive task when performed without specified methods and goals. The training process involves finding a machine learning algorithm that works well with the input data matrix. For the scope of this thesis, it was decided to solely rely on classifiers made available by the sklearn python library. Especially, since the reliability assessment for individual predictions and the connected perturbation approach is the centre of the thesis, simple and robust models were preferred. For general model selection throughout multiple iterations, the evaluation metrics which have been considered were (i) *accuracy*, (ii) *precision*, (iii) *recall* and (iv) *f1-score*. Accuracy is to be regarded as the most important metric, with the others leading to decisions according to overall good performance, as indicated by f1 scores or particularly good precision or recall values. After multiple iterations of the CRISP-DM process, the choice of which models are used for later reliability assessment fell into the three ensemble methods namely (i) *Random Forest Classifier*, (ii) *Adaptive Boosting Classifier* and (iii) *Gradient Boosting Classifier*. All of which are tree-based classifiers, which are suitable for a wide variety of data types and data structures. Additionally, scaling and

normalisation of training and test datasets is not needed, making the models easier to handle throughout all stages of the CRISP-DM process.

Figure 3.15 shows all steps that were taken within the modelling process. The process starts with loading the prepared dataset as described in previous sections from the stored pickle files. This dataset is then split to obtain a training and test dataset. The test dataset is put aside for the whole training process. Because of the high frequency of on-time arrivals compared to late and early arrivals, sampling techniques are required for many machine learning models that struggle with classification on imbalanced datasets. Furthermore, especially distance-based ML algorithms require scalers for optimal training results. Therefore, after evaluating the characteristics of each algorithm, a decision is made on whether scaling and sampling is needed. Although beneficial to most tried out models, the three remaining tree algorithms did not perform better with sampling methods, which ultimately led to the decision to neglect sampling and scaling for the used models altogether. However, in cases where scaling was mandatory, the scaler needs to be saved to transform test data into the correct feature space, which also has to be considered when altering input vectors for perturbation. After eventual preprocessing, the next steps are comprised of finding optimal model hyperparameters for each model type by initially using *Successive Halving* [42] to identify the most important parameters for each model in a computationally cheaper way. *Successive Halving* is also used to identify a good region within the parameter space around which the model promises to deliver results. After initial parameter estimation, the fine-tuning strategy *Exhaustive Grid Search* is used to finalize the parameter set, by looking through all combinations around the previously identified feature space. Due to the combinatory complexity that comes with *Exhaustive Grid Search*, the process is computationally expensive, making the limitation for a

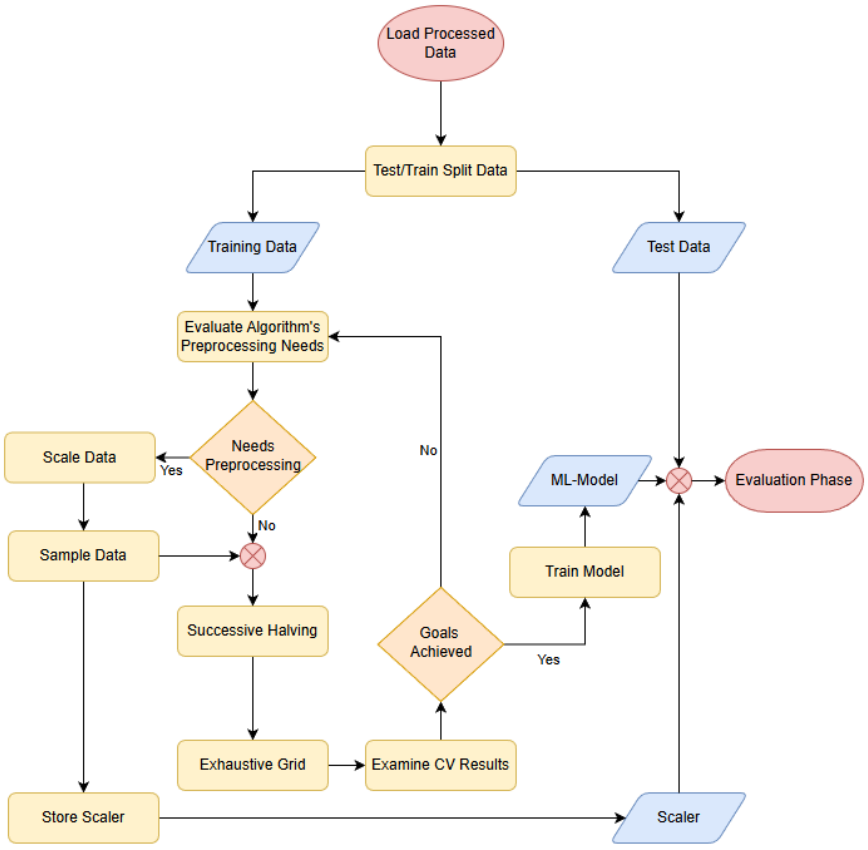


Figure 3.15: Flowchart of the modelling process.

local area for most important parameter ideal to safe resources. Results of parameter fine-tuning were generally evaluated with a 10-fold cross validation targeting accuracy values for selecting the best parameter set. After fine-tuning, the models' cross validation results are evaluated with defined baseline limits in mind, which are explained in detail in the following section. The final parameters of the used classifiers are shown in Table 3.10. For each parameter not shown in the table, the default parameter as provided by the sklearn library is used. The '—' character indicates parameters that are not available for the corresponding classifier. Lastly, the model is trained on the whole training dataset, depending on the identified parameter set. Finally, the training process itself is deemed successful, resulting in the model's compression and storage in the project workspace as *'joblib'* file ready for evaluation and eventually, deployment.

Table 3.10: Final parameters for each classifier instance.

Parameter	**Classifier**		
	Random Forest	**Adaptive Boosting**	**Gradient Boosting**
n_estimators	510	720	600
max_depth	20	—	7
max_features	0.3	—	'null'
max_samples	'null'	—	—
class_weight	'balanced'	—	—
learning_rate	—	1.1	0.1

3.5 Evaluation

CRISP-DM defines the evaluation stage of a machine learning process as being independent of the modelling phases [8]. By defining baselines and comparing created models to these baselines and each other through particularly selected evaluation metrics, the decision to push a given artifact, in the sense of a trained ML model, into deployment phases can be made. Insufficient model performance might introduce the need to reevaluate past design decisions by starting a new iteration of the CRISP-DM process. This section describes the process of finding metrics used for model evaluation and explains the characteristics of the proposed metrics. Then, the baseline models are explained in detail to show the minimal requirements for the created machine learning's performance. Lastly, the performances of the final models are shown and compared to the targets declared by the baseline strategy.

3.5.1 Metrics

This thesis consists of a classification problem for carrier delays regarding the three target labels describing early, on-time and late arrivals that have been proposed throughout this thesis. To evaluate the performances for each selected model regarding the underlying classification problem, (i) *accuracy score*, (ii) *precision score*, (iii) *recall score* and (iv) *f1 score* were used. All metric implementations were taken from sklearn and are used to compare the models by evaluating metrics over all data entries put aside for testing in the test set. *Accuracy* describes the percentage of all correctly classified test cases over all possible scenarios, *precision* describes the percentage of all

correctly classified test case over all identified relevant cases and *recall* describes the percentage of identified cases over all relevant cases in the test set. Typically, *precision* and *recall* have an inverse relationship to each other. Therefore, *f1-score* introduces a metric that is calculated by forming the harmonic mean of precision and recall, allowing for a balanced comparison between models with different characteristics regarding *precision* and *recall*. Since this thesis handles a multi class prediction problem consisting of three possible target values, a given predicted target competes with an additional possible outcome compared to binary classification. To compare the models' performances over all classes, an averaging method needs to be defined over all class performances. This can be done by using either *macro* or *micro* precision and recall metrics. A macro metric is calculated by looking at each possible class in a one vs all manner, averaging these results over all classes. It has to be noted, that imbalanced data structures influence the calculation. In such cases, a weighted approach could turn out to be beneficial. For this thesis, simple macro averages without weighting are used. Although this thesis is interested in the performance over all classes equally, the most occurring class *'on time'* is also the most critical, making macro values suitable for this scenario. In multi class problems, micro averaging results in having equal values for precision and recall, which turns out to be equal to global accuracy values. Therefore, only macro averaged values are displayed in the further scope of this thesis.

3.5.2 Baselines

To find a decision regarding finishing the process of modelling within the CRISP-DM framework, an evaluation strategy is needed to define a threshold that a given model has to overcome. For this thesis, two baselines were established to formulate targets for varying degrees of confidence in the model. The first baseline *'Majority Guessing'* assumes a model that always predicts every flight to be on-time and must be beaten by the model at both *accuracy* and *f1-score*. The second baseline *'Departure Delay Inference'* uses the departure delay of a given flight and assumes that the flight arrives with the same delay. Classifying the delay according to the logic of the FAA, as shown in Equation 3.2. Departure delay is such a critical feature in the training dataset, that many machine learning projects for flight delay prediction show a strong influence of the feature on predictive capabilities. Furthermore, the importance of this feature is shown within the scope of this thesis by correlation analysis and feature importance comparisons. Due to this relevance of the feature, the baseline defined is a solid minimum requirement for which some models could even potentially fail to meet the *f1-score*. A model failing against baseline *f1-score* might still be accepted into later stages of the process, depending on its performance over the various classes. However, a machine learning model must outperform the *'Departure Delay Inference'* baseline by a significant margin in accuracy metrics over all target classes.

Majority Guessing

Using majority guessing as a baseline is a common procedure to compare machine learning metrics in imbalanced machine learning tasks at a very fundamental level. Figure 3.16 demonstrates the confusion matrix of this baseline, showing again that all cases are predicted as on-time, leading to a perfect classification for all truly on-time flights while naturally completely misclassifying both early and late arrivals. Table 3.11 shows all performance metrics regarding this baseline model in the column *BL-MG*. The *Majority Guessing* baseline results in an *accuracy* and *precision* for

on-time flights of both 56.71%, which comes from the dataset's class distribution property. The *macro f1 score* of 24.12% is a mark that no model should struggle to beat. This process was established in early stages when departure delay was not yet used in the training dataset. As described, this baseline forms a hard threshold for all models to overcome. Models that cannot overcome this target accuracy are scrapped immediately.

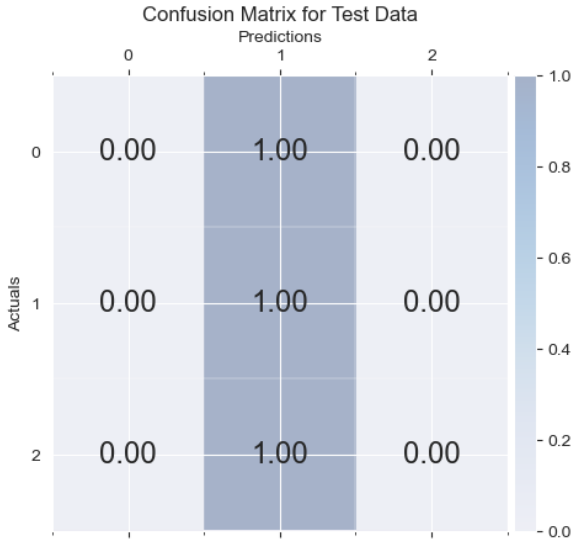


Figure 3.16: Confusion matrix for Majority Guessing baseline on test data.

Departure Delay Inference

Since the *'Majority Guessing'* baseline, with an accuracy metric of 56.71%, shows a very low hurdle to overcome, it was favourable for the outcome of machine learning development to come up with a new baseline that requires a higher standard from the model's performance. In early iterations of the CRISP-DM cycle, the feature for departure delays was not used due to it being overly important and possibly skewing the algorithm's focus to solely rely on this one feature. These iterations and other projects using similar data like, Truong et al. [34] and Hu et al. [35] however showed, that discarding departure delay hinders the model's performance significantly. Because of this insight, later experiments were conducted, which showed that departure delay is by far the strongest descriptor in the dataset. To then be able to distinguish the created model's performance from the single impact of the departure delay feature, the *'Departure Delay Inference'* baseline was constructed. Therefore, if a given model beats the baseline which solely relies on departure delay, one can assume that the other features combined with the model's parameters have enough of a positive impact on the overall classification task to conclude a successful modelling phase.

Looking at the evaluation metrics for the *'Departure Delay Inference'* baseline, the results look as shown in Table 3.11 in the column described as *BL-DI*. We can observe that a classifier which only uses departure delay to infer arrival delay does perform reasonably well over all classes, achieving an *accuracy* of 64.71%, *macro precision* of 75.08%, *macro recall* of 57.56% and *macro f1-score* of 51.12%. Especially on-time arrivals have a high *accuracy* with 94% indicated in the confusion matrix in Table 3.17. Late flights can also be identified reasonably well, with an accuracy of 78%. However, hardly any early arrivals have been identified correctly by this baseline. By following this baseline strategy, only 193 early departures within the test dataset are correctly identified, while 20752 flights are registered to actually arrive early at the destination airport. This imbalance from

Figure 3.17: Confusion matrix for Departure Delay Inference baseline on test data.

early departures to early arrivals is no characteristic reserved to the test set, it can also be found within the training dataset. The confusion matrix displayed in Figure 3.17 further indicates the shortcomings of such a simple method that relies solely on departure delay. When it comes to on-time arrivals or late arrivals, a simple departure delay inference approach might show decent enough performance for everyday passengers, making this approach a solid baseline to aim for. For models within this thesis, the metrics to beat are the *accuracy* and *macro precision* metrics. Due to the imbalanced dataset, bias towards the majority class and importance of the departure delay feature to the general classification task, this baseline model shows to be a strong hurdle and is therefore deemed to be enough to justify a conclusion of the modelling processes for a given model that beats the baseline metrics.

3.5.3 Trained Models

By following CRISP-DM and iteratively improving decisions based on the learnings of previous phases, three machine learning models were created and evaluated against each other. The models' algorithms, namely *Random Forest Classifier*, *Adaptive Boosting Classifier* and *Gradient Boosting Classifier*, were taken from the sklearn python library and trained with the data as described in previous sections. The models are further referenced based on the corresponding machine learning algorithm used. In this section, the success of a given model is evaluated in comparison to the previously explained baselines. Furthermore, strengths and weaknesses are indicated for each model type. For all models, the majority guessing baseline needs to be surpassed for all macro metrics and accuracy, while the delay inference baseline must be surpassed in accuracy to classify a training approach as successful. Finally, all trained models are compared to each other to find key differences within the model's respective performances.

Random Forest

Because of its robustness, the random forest algorithm was the first model to show good performance metrics, beating baselines with parameter optimisation on suboptimal datasets. Furthermore, due to the built-in weighting parameter, it showed reasonably balanced results over all target variables

as indicative by the confusion matrix shown in Figure 3.18. With a global accuracy of 73.43% and decently balanced precision and recall metrics over all single-label metrics shown in Table 3.11, the *Random Forest Classifier* beat the *Departure Delay Inference* baseline on all macro baselines. On the less important class-specific metrics, the *Random Forest Classifier* also outperformed the baseline on nearly all measures. It only fell short on the unfavourable metrics, being precision of early arrivals, recall of on-time arrivals and recall of late arrivals. With a macro f1-score of 73.03%, the difference to the baseline value is still significant, making the model pass the requirements in all critical metrics. It is noteworthy that hardly any true early or on-time arrivals are wrongly classified as late arrivals. This combined with the macro precision of 88.78% shows that the *Random Forest Classifier* is especially competent at recognizing late arriving aircraft, which is often times the most problematic case for operators.

Figure 3.18: Confusion matrix for Random Forest model on test data.

Adaptive Boosting (AdaBoost)

Adaptive boosting was tried out to evaluate a boosting technique contrary to the robust *Random Forest Classifier* ensemble method. While not being as precise on the early arrivals and late arrivals, it was able to outperform both baselines with an accuracy score of 71.45% and a macro f1-score of 68.26%. As Figure 3.19 shows, 86% of on-time flights were correctly identified, which beats the *Random Forest Classifier* in that regard while losing to it by 4% on late arrivals. The most significant difference for evaluation metrics between the random forest classifier and the *Adaptive Boosting Classifier* is observed for early arriving aircraft where the *Adaptive Boosting Classifier* only recognized 42%, which pales in comparison to the *Random Forest Classifier's* 64% accuracy on early arrivals.

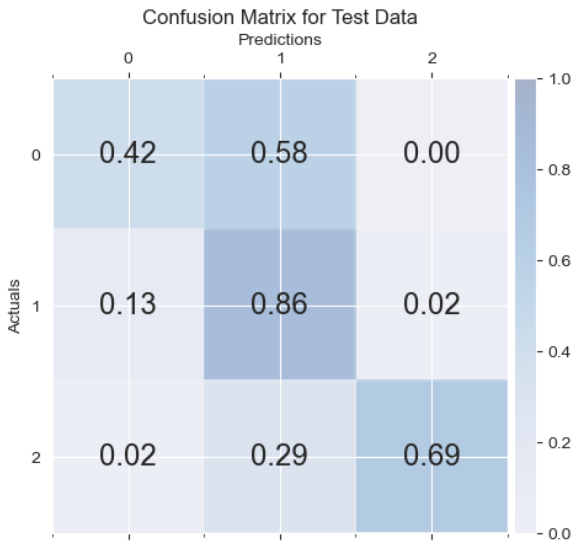


Figure 3.19: Confusion matrix for Adaptive Boosting model on test data.

Gradient Boosting Classifier

The last model which showed promising results after training stages was the *Gradient Boosting Classifier*. The confusion matrix depicted in Figure 3.20 shows similar patterns to the confusion matrix of the *Adaptive Boosting Classifier* shown in Figure 3.19, as the model classifies on-time arrivals best, late arrivals second best and showing weaknesses on classification of early-arrivals. Similarly to the previously shown model metrics, Table 3.11 shows all final metrics in the corresponding column named *'XGB'*. Contrary to the *Adaptive Boosting Classifier*, the *Gradient Boosting Classifier* outperformed the *Random Forest Classifier* with an accuracy of 75.19%. Due to the worse characteristics for early-arriving flights, the gradient boosting's *macro f1-score* of 72.97% is trailing 0.06 percentage points behind the *macro f1-score* of the random forest. Overall, the *Gradient Boosting Classifier* performs best on most metrics, only missing the *Random Forest Classifiers* by a couple of percentage points in the categories where it lost out.

Figure 3.20: Confusion matrix for Gradient Boosting model on test data.

Conclusion of Evaluation

Table 3.11 shows all collected metrics for each of the selected algorithms and the baselines to allow for fast comparisons between the models. For spatial reasons, model and baseline names were abbreviated according to the following scheme. BL-MG indicates the majority guessing baseline, BL-DI the departure delay inference baseline, RF the Random Forest Classifier, AdaBoost the Adaptive Boosting Classifier and XGB the Gradient Boosting Classifier. For each metric concerning the trained models, the best value is marked with a green background to allow easier identification of strengths and weaknesses for each model. Most notably, the Adaptive Boosting Classifier does not win in any category that has been observed. Between the Gradient Boosting Classifier and the Random Forest Classifier, the first is more capable of correctly identifying the minority classes, while the latter partially makes this deficiency up by finding more relevant incidents, which is indicative by its higher recall score. Generally, the differences between the Random Forest Classifier compared to the Gradient Boosting Classifier are not strong enough in any part that the Random Forest Classifier dominates, that it would warrant the selection of the Random Forest Classifier over the Gradient Boosting Classifier for the deployment phase. Although all classifiers' predictions are further assessed with the perturbation approach described in the thesis, in a theoretical scenario in which one of the models is selected for deployment, the Gradient Boosting Classifier would have been chosen.

Table 3.11: Comparison of evaluation metrics over baselines and models.

Metric	BL-MG	BL-DI	RF	AdaBoost	XGB
Global Test Measures					
Number of Entries	65801	65801	65801	65801	65801
Accuracy	56.71%	64.71%	73.43%	71.46%	75.19%
Precision-M	18.90%	75.08%	75.10%	73.46%	77.08%
Recall-M	33.33%	57.56%	71.57%	65.31%	70.24%
F1-Score-M	24.12%	51.12%	73.03%	68.26%	72.97%
Test Measures for Label '0'					
Number of Entries	17839	17839	17839	17839	17839
Precision Score	—	87.05%	58.30%	59.07%	65.86%
Recall Score	—	00.81%	63.96%	41.54%	51.77%
Test Measures for Label '1'					
Number of Entries	39370	39370	39370	39370	39370
Precision Score	56.71%	62.23%	78.22%	72.30%	75.88%
Recall Score	100%	94.24%	77.84%	85.60%	86.36%
Test Measures for Label '2'					
Number of Entries	8592	8592	8592	8592	8592
Precision Score	—	75.97%	88.78%	89.01%	89.50%
Recall Score	—	77.64%	72.91%	68.80%	72.59%

4

Perturbation Evaluation

Perturbation describes the process of altering input data by small margins based on findings in the data understanding or data preprocessing phases of CRISP-DM. This chapter introduces the declared perturbation options based upon these findings and explains the logic and reasoning behind their declaration for each option. Additionally, the creation of the perturbation dataset is described in detail to establish a foundation from which further analysis is performed. Lastly, findings and metrics are established to inform about potential benefits of using the perturbation approach on this particular use-case.

4.1 Perturbation Options

The options defined for this thesis are derived from insights into the dataset's characteristics. Generally, perturbation options are formulated by looking at properties from input features that are subject to possible volatilities within the dataset. For example, one feature might hold information gathered by physical sensors, which are influenced by measurement uncertainty. Other examples of possible perturbation options connected to data preprocessing are features that have been binned or estimated in order to improve data quality and model performance. All features that are suspect of any sort of inherent uncertainty should, in theory, not allow a change of target prediction when deviating from the originally measured value within these margins. For all suitable features, these margins are documented in corresponding perturbation options and used to later create deviating data entries as model input within these margins for which the predicted outcome should theoretically not change from the original data entry's prediction outcome.When a machine learning model then produces deviating predictions on a data entry that is covered by a perturbation option, the original prediction might be based on an edge case in the data or a sensitive area near the model's decision boundary, deeming the original prediction outcome unreliable. Features that have been binned or estimated and features with underlying measuring uncertainties are indicated by red perturbation options as defined by Staudinger et al. [6]. Features whose margins are broad estimations based on some characteristic of the data, but not defined as precisely as the margins for features connected with red perturbation options, are indicated by orange perturbation options. Other features that have some potential for perturbation, but with drastic margins, which expect a change of target classification, are defined with green perturbation options. The choice of classifying a perturbation option as green or yellow is therefore not as clear-cut. Ultimately, this decision is made by development teams and is based on the feature importance for the model

prediction and also the importance of the feature for stakeholders.

The preceding chapters introduced the data mining process related to the use case for this thesis. Since the evaluation of the perturbation process is analysed with regard to model predictions within Python scripts, the categorical perturbation option definitions with the labels of *red*, *orange* and *green* were suboptimal for conditional operations. Therefore, within this thesis, perturbation levels are described numerically, changing red options to *level 3*, orange options to *level 2* and green options to *level 1* options.
The following section shows all used perturbation options and their corresponding algorithm for creating the deviating data entries. The perturbation options are grouped based on their perturbation levels, such that *level 3* options are described first, with *level 1* options being described last. Within the groups, the relevant perturbation options are explained one after the other. Each option is also explained in tabular form, showing all necessary attributes of each perturbation option. The attributes shown within the option declaration tables are (i) feature name, (ii) parameter for the perturbation method, (iii) perturbation level, (iv) CRISP-DM phase on which the option is based and (v) the algorithm used to generate the deviating data entries with respect to the input parameter.

4.1.1 Algorithms for Perturbation

The perturbation approach is based on the analysis of differences in classification outcomes based on marginal differences in input vector features. Since one could not only perturb single features, but also the effects of alterations on a combination of features, the resulting dataset containing the perturbed input vectors would grow exponentially with the number of feature combinations. As this would lead to an unfeasibly high time-complexity, it was chosen to look into single-feature perturbations only. Therefore, perturbation options are declared such that each feature can only be described by one option, which also holds information about the algorithm used to calculate deviations of the original input vector. To calculate the perturbed feature vectors, three methods are used, depending on the best fit for each feature. These methods consist of perturbing by (i) percentage (pertPerc), (ii) absolute value (pertAbs), or (iii) either absolute or relative value, depending on whichever value is greater (pertPercOrAbs). For the latter, the appropriate method is selected by comparing the maximum possible deviation from the original value. In this section, both possible algorithms are further explained in detail.

Perturbation by Percentage

When a declared perturbation option within the scope of this thesis declares a feature to be perturbed by *'pertPerc'* each input value is altered between a lower and upper bound limit evenly spaced with the original value at its centre. The upper and lower bounds are the maximum deviations from the original value, depending on the $perc$ parameter, as showcased in Algorithm 1. Between the upper and lower limit, 8 evenly spaced data points are additionally added by adding and subtracting multiples of the calculated step size from the original input value to allow a more fine-tuned perturbation analysis.

Algorithm 1 Perturbation by Percentage

function $pertPerc(val, perc)$
  $steps \leftarrow 4$
  $lower\_bound \leftarrow val \times (1 - \frac{perc}{100})$
  $upper\_bound \leftarrow val \times (1 + \frac{perc}{100})$
  $step\_size \leftarrow \frac{upper\_bound - lower\_bound}{2 \times steps}$
  $arr \leftarrow []$
  for $i \leftarrow -steps$ **to** -1 **do**
    $arr \leftarrow [\ldots, val + i \times step\_size]$
  end for
  for $i \leftarrow 1$ **to** $steps$ **do**
    $arr \leftarrow [\ldots, val + i \times step\_size]$
  end for
  return arr
end function

Perturbation by Absolute Value

In contrast to perturbation by a given percentage value, some features are provided resolutions or absolute value deviations, within which one can expect volatility for the corresponding feature. For this case, a given input value is altered by absolute margins according to Algorithm 2. Here it shows that first, an upper and lower bound is created by adding and subtracting the possible absolute value deviation from the original value. Between these boundaries, 8 evenly spaced values are created, which are then used for classification and further perturbation analysis. The process is therefore very similar to the perturbation by percentage. One notable difference is the usage of a rounding variable, which represents the number of decimal places needed to preserve the granularity of the perturbed value. This avoids unnecessarily long floating-point representations and ensures consistency in the output. The rounding variable is passed to Python's *round(current_value, rounding)* [50], which rounds each *current_value* to the number of decimal places specified by the

Algorithm 2 Perturbation by Resolution

function $pertResolution(val, res)$
  $steps \leftarrow 4$
  $lower\_bound \leftarrow val - res$
  $upper\_bound \leftarrow val + res$
  $step\_size \leftarrow \frac{res}{steps}$
  $rounding \leftarrow \max(0, -\lfloor \log_{10}(step\_size) \rfloor)$
  $arr \leftarrow []$
  for $i \leftarrow -steps$ **to** -1 **do**
    $current\_value \leftarrow val + i \times step\_size$
    $rounded\_value \leftarrow \text{round}(current\_value, rounding)$
    $arr \leftarrow [\ldots, rounded\_value]$
  end for
  for $i \leftarrow 1$ **to** $steps$ **do**
    $current\_value \leftarrow val + i \times step\_size$
    $rounded\_value \leftarrow \text{round}(current\_value, rounding)$
    $arr \leftarrow [\ldots, rounded\_value]$
  end for
  return arr
end function

rounding parameter of the function.

4.1.2 Definition of Perturbation Options

This section mostly captures features within the input dataset that were directly taken from METAR dataset. This comes down to the characteristics that most METAR features are derived from sensor output and are therefore subject to uncertainties and errors from said sensors. As described in Section 4.1, features based on sensor data are prime examples for *level 3* perturbation options, which is also the most occurring perturbation level for the options defined in this thesis. Besides METAR data, other noteworthy data features are aircraft dimension features and the runway availability feature derived from NOTAM reports. The definitions and explanations for each *level 2* and *level 3* perturbation option is described in the following section.

Level 3 Options

When it comes to the data understanding stage, most derived perturbation options come from sensors that were used to measure the given value for the dataset. Since measuring physical units comes with uncertainties and physical limits for the sensors, we can never be sure that a given sensor data point at a given time is truly the actual physical value. Therefore, manufacturers inform users about expected errors, resolutions, data ranges and sensor tolerances. In this thesis, the ASOS User Guide [29] informs us exactly about these values. Since these uncertainties are prime examples for *level 3* perturbation options, the values defined in the ASOS User Guide are used to base the option definitions on. Although the ASOS User Guide documents the corresponding precisions, resolutions and data ranges for each sensor, the generational algorithms shown in Algorithms 1 and 2 are kept rather simple and generic. Therefore, the perturbation options have to be defined such that the parameters derived from the tolerances are usable as simple function inputs for the generating algorithm. Since tolerances are often non-linear, it is necessary to make design decisions in order to best represent the underlying tolerances with the generic generational algorithm and their corresponding input parameters. Therefore, each perturbation option is defined separately with respect to the particular needs each data feature has. The following section explains all design decisions made for each of the level 3 perturbation options and explains the basis on which they are formulated on.

The first two perturbation options explain the perturbation logic for the data features *'TEMP(C)'* and *'DEWPOINT_TEMP(C)'* which are shown in Table 4.1. Both options are declared based on the ASOS User Guide's documented root mean squared error and their corresponding temperature ranges, as shown in Table 4.2. According to the ASOS User Guide [29], both, ambient temperature and dew point temperature are measured directly with differing RMSE for each specific temperature range shown in Table 4.2. Since the generating algorithms used in this thesis are formulated rather generically, meaning that they are only capable of representing a deviation by percentage or absolute value of a single data value, one of the RMSE values had to be chosen as input parameter depending on the corresponding temperature range it covered. For both ambient temperature and dew point temperature, it was chosen to select the RMSE, which accounts for the median temperature of the respective feature in the processed training dataset. For ambient temperature, we identified a median temperature of $19.39°C$ $(66.9°F)$, while dew point temperature comes with a median of $13.89°C$ $(58°F)$ within the processed dataset. According to Table 4.2, we expect

Feature Name	TEMP(C)
Parameter	1.345
Level	3
Phase of Declaration	Data Understanding
Generating Algorithm	Pert_Percent
Feature Name	DEWPOINT_TEMP(C)
Parameter	1.93
Level	3
Phase of Declaration	Data Understanding
Generating Algorithm	Pert_Percent

Table 4.1: Option declaration for features TEMP(C) and DEWPOINT_TEMP(C)

an error of $0.9°F$ for each measurement of ambient temperature. Dew point temperature's RMSE for the range that represents the median value is also given as a range value that fluctuates strongly from $1.1°F$ to $4.7°F$. Since previous data understanding processes showed that dew point temperature is an important feature for the used classifiers, a change of $4.7°F$ for each value is too drastic of a change for the input value. Therefore, it was chosen to select the lower bound of the RMSE range. Furthermore, the ranges of RMSE are given in degree Fahrenheit and not Celsius, which is the metric chosen to represent the dataset. To normalise the impact of a perturbation within the described error margin, the Root Mean Squared Error in degrees Fahrenheit (RMSE_F) is expressed relative to the median temperature in the dataset (T_C), converted to Fahrenheit. This normalization provides insight into how significant the RMSE is compared to the typical temperature range, resulting in a percentage value usable for the perturbation option's definition. The relative error percentage ($\text{RMSE}_{F,\text{rel}}$) is computed as shown in Equation 4.1 and shows the maximum deviation according to RMSE at the median temperature in percentage.

$$\text{RMSE}_{F,\text{rel}} = \left(1 - \frac{(T_C \cdot 1.8 + 32) - \text{RMSE}_F}{T_C \cdot 1.8 + 32}\right) \cdot 100 \tag{4.1}$$

Equation 4.1 applies for both dew-point temperature and ambient temperature. With T_C of

Parameter	**Range**	**RMSE**	**Max Error**	**Resolution**
Ambient Temperature	-80°F to -58°F	1.8°F	±3.6°F	0.1°F
	-58°F to +122°F	0.9°F	±1.8°F	
	+122°F to +130°F	1.8°F	±3.6°F	
Dew Point Temperature	-80°F to -0.4°F	3.1°F to 7.9°F	4.5°F to 13.9°F	0.1°F
	-0.4°F to +32°F	2.0°F to 7.9°F	3.4°F to 13.9°F	
	+32°F to +86°F	1.1°F to 4.7°F	2.0°F to 7.9°F	

Table 4.2: Temperature measurement accuracy and resolution. Adapted from [29].

$19.39°C$ and RMSE_F of $0.9°F$ for ambient temperature and T_C of $13.89°C$ and RMSE_F of $1.1°C$ for dew-point temperature. Equation 4.1 yields the relative error magnitudes ($\text{RMSE}_{F,\text{rel}}$) of 1.345% and 1.93% for ambient temperature and dew-point temperature, respectively. Ultimately, these values are used for the parameter of both perturbation options, as seen in Table 4.1.

Table 4.3 shows the final version for the perturbation option concerning the data feature WIND_SPEED(KMH). According to the ASOS User Guide [29], wind speed is measured in two-minute averages of measurements that are being collected every 5 seconds. These rolling two-minute averages are then transmitted to the ASOS at one-minute intervals. Taking sensor tolerances and aggregation into account, ASOS reports a measurement accuracy of ± 2 knots or 5% within a range of 0–125 knots. 125 knots, or 231.5 km/h in metric units, is not a relevant upper bound for this attribute, since there is no data entry exceeding 70 km/h and there will hardly be any carrier at such high ground-level wind speeds. For accuracy values, ASOS states it to be either ± 2 knots or 5% of a sensor's measurement, depending on whichever value is greater. Since 2 knots equal around 3.704 km/h, the final perturbation option for wind speed looks as described in Table 4.3.

Feature Name	WIND_SPEED(KMH)
Parameter	3.704 or 5%
Level	3
Phase of Declaration	Data Understanding
Generating Algorithm	Pert_Percent_or_Res

Table 4.3: Option declaration for feature WIND_SPEED(KMH).

Also related to wind sensors is the data attribute WIND_DRCT(DEG) which describes the wind direction in degrees relative to north. The ASOS user guide references that, similarly to wind speed measurements, wind direction is measured every five seconds and presented within a two-minute rolling average that is updated every minute. A collected data point is then rounded to the nearest 10-degree increment, which results in an integer perturbation range of ± 5 degrees as shown in Table 4.4.

.

Feature Name	WIND_DRCT(DEG)
Parameter	5
Level	3
Phase of Declaration	Data Understanding
Generating Algorithm	Pert_Res

Table 4.4: Option declaration for feature WIND_DRCT(DEG).

Relative humidity is calculated based on ambient temperature and dew point temperature given by the respective sensors. The perturbation logic for relative humidity is designed to account for the maximum error impact due to tolerances in ambient temperature and dew point temperature

measurements. In order to find a strict margin that is usable within the perturbation option, an error propagation approach was chosen based on a simplified formula for relative humidity [51]:

$$f(T_a, T_d) = 100 - (T_a - T_d) \cdot 5 \tag{4.2}$$

where:

- T_a is the measured ambient temperature in degrees Celsius.
- T_d is the measured dew point temperature in degrees Celsius.
- $f(T_a, T_d)$ is the relative humidity in percent.

This simplified formula does not produce exact results for relative humidity with respect to ambient temperature and dew point temperature, but provides sufficient results within the 50% to 100% relative humidity bandwidth, which is the range that covers the majority of data values in the training dataset. Furthermore, due to the function's linear characteristics, it is much easier to calculate its partial derivatives and allows us to use error propagation in a more accessible way.
As previously seen in Table 4.2, the root mean squared error (RMSE) of the ambient temperature is $0.9°F$ $(0.5°C)$ while the RMSE for dew point temperature amounts to $1.1°F$ $(0.611°C)$. To stay consistent with previous assumptions, the same temperature intervals for RMSE are used as described in the feature's respective perturbation options. By using the RMSE of both temperature measurements, the effects of error propagation on the resulting relative humidity calculations can be derived, as shown by Lin et al. [52]. This process yields the standard uncertainty u, which represents the standard deviation of the distribution that results from error propagation. To calculate u, the RMSE of the input temperature values are assumed to be based on normally distributed data, which equals both RMSE values to be the standard uncertainty for the respective temperature distributions $u(T_a)$ and $u(T_d)$. Although the process of Lin et al. is followed due to the similarity to the problem, contrary to their work, the expanded uncertainty [53], which can be used to cover a wider confidence interval for the calculated standard uncertainty, is not used in this thesis. Therefore, it is assumed that $k = 1$. The standard uncertainty of the derived relative humidity is then used as a parameter for the perturbation option regarding the relative humidity feature. The formula to calculate the standard uncertainty u is as follows [52]:

$$u = k \cdot \sqrt{\left(\frac{\partial}{\partial T_a} f(T_a, T_d)\right)^2 \cdot u^2(T_a) + \left(\frac{\partial}{\partial T_d} f(T_a, T_d)\right)^2 \cdot u^2(T_d)} \tag{4.3}$$

where:

- $u^2(T_a)$ and $u^2(T_d)$ are the squared standard uncertainties (variance) for ambient and dew point temperature.
- u is the standard uncertainty for derived relative humidity.
- k is the coverage factor for the expanded uncertainty.

Solving partial derivatives for Equation 4.3 as follows:

$$\frac{\partial}{\partial T_a} f(T_a, T_d) = \frac{\partial}{\partial T_a}(100 - (T_a - T_d) \cdot 5) \tag{4.4}$$

$$\frac{\partial}{\partial T_a} f(T_a, T_d) = \frac{\partial}{\partial T_a}(100 - 5T_a + 5T_d) \tag{4.5}$$

$$\frac{\partial}{\partial T_a} f(T_a, T_d) = -5 \tag{4.6}$$

and

$$\frac{\partial}{\partial T_d} f(T_a, T_d) = \frac{\partial}{\partial T_d}(100 - (T_a - T_d) \cdot 5) \tag{4.7}$$

$$\frac{\partial}{\partial T_d} f(T_a, T_d) = \frac{\partial}{\partial T_d}(100 - 5T_a + 5T_d) \tag{4.8}$$

$$\frac{\partial}{\partial T_d} f(T_a, T_d) = 5 \tag{4.9}$$

Inserting the resulting values and assumed coverage factor back into Equation 4.3:

$$u = 1 \cdot \sqrt{(-5)^2 \cdot (0.5)^2 + 5^2 \cdot (0.611)^2} \tag{4.10}$$

$$u = 3.948 \tag{4.11}$$

Obtaining the amount of change possible in relative humidity based on the RMSE of temperature values. In order to stay consistent with previously defined perturbation options, it is observed by how much of a margin in percentage a change of 3.948 would bring for median temperature values. Therefore, inserting median temperature values into the following formula obtains the change in relative humidity in percent at median temperatures as follows:

$$\frac{u}{100 - (Ta_{median} - Td_{median}) \cdot 5} \cdot 100 = u_{perc} \tag{4.12}$$

$$\frac{3.948}{100 - (19.389 - 13.889) \cdot 5} \cdot 100 = 5.445 \tag{4.13}$$

Finally, the perturbation option is defined by using either uncertainty of the derived of relative humidity u or the change in percentage, u_{perc} whichever is greater as seen in Table 4.5.

Feature Name	REL_HUMIDITY(PERCENT)
Parameter	3.948 or 5.445%
Level	3
Phase of Declaration	Data Understanding
Generating Algorithm	Pert_Percent_or_Res

Table 4.5: Option declaration for feature REL_HUMIDITY(PERCENT).

The three data features' *'1HOUR_PRECIPITATION(INCH)'*, *'VISIBILITY(MILES)'* and *'SEA_LEVEL_PRESSURE(MILLIBAR)'* parameter values for the perturbation option definition are simply derived from the ASOS User Guide and did not necessitate further adjustments. These three option declarations are shown in Table 4.6. The data attribute for sea level pressure is described to have an accuracy of ± 0.02 inches of mercury, which equates to an air pressure differential of 0.7 millibars. Visibility sensors have increasing uncertainty with increased visibility, ranging from $\pm\frac{1}{2}$mile uncertainty at the range from 0 miles to $1\frac{1}{2}$ miles, going up to ± 1 mile of uncertainty at 4 miles or above with an 80% confidence interval. Since the lower end of visibility is more critical to the aviation use case, it was chosen to use the $\pm\frac{1}{2}$ mile uncertainty for the corresponding perturbation option. Lastly, precipitation is measured in inches and is documented to have a resolution of 0.02 inches, which is used to alter the input value within this margin as defined in the perturbation option.

Feature Name	SEA_LEVEL_PRESSURE(MILLIBAR)
Parameter	0.7
Level	3
Phase of Declaration	Data Understanding
Generating Algorithm	Pert_Res
Feature Name	VISIBILITIY(MILES)
Parameter	0.25
Level	3
Phase of Declaration	Data Understanding
Generating Algorithm	Pert_Res
Feature Name	1HOUR_PRECIPITATION(INCH)
Parameter	0.02
Level	3
Phase of Declaration	Data Understanding
Generating Algorithm	Pert_Res

Table 4.6: Combined option declarations for features 1HOUR_PRECIPITATION(INCH), VISIBILITIY(MILES) and SEA_LEVEL_PRESSURE(MILLIBAR).

The final *level 3* perturbation option defined for this thesis is the only *level 3* feature not originating from weather data, namely *'RUNWAY_ERROR(PERC)'*. This feature holds information regarding the availability of runways as reported by NOTAM messages. The feature was constructed during data preprocessing stages and can only take values from 0 to 1 in steps of 0.2 which is used to formulate the perturbation option as seen in Table 4.7. Since no values beneath are possible due to the fixed number of runways on the ATL airport.

Feature Name	RUNWAY_ERROR(PERC)
Parameter	0.2
Level	3
Phase of Declaration	Data Preparation
Generating Algorithm	Pert_Res

Table 4.7: Option declaration for feature RUNWAY_ERROR(PERC).

Level 2 Options

Features covered with *level 2* options possess some characteristic fit for perturbation, where margins for perturbation are not as precisely defined as sensor tolerances, bins, or estimation errors as seen for *level 3* options. Within the scope of this thesis, three features qualified for *level 2* perturbation options, which all stem from the aircraft dataset. Since all of these features inherit the same characteristic of having fixed values for a given aircraft type, the same process to identify fitting perturbation margins was applied to each feature derived from the aircraft dataset. Figure 4.1 shows the structure of the aircraft dimension features on the examples of the features *'TAIL_HEIGHT(M)'*, *'APPROACH_SPEED(KMH)'* and *'PARKING_AREA(SQM)'*. The y-axis shows the absolute frequency for each unique value in the dataset. Since some aircraft are overrepresented, the y-axis is shown in logarithmic scale. Since a given aircraft type's dimensions are always strictly defined in the aircraft's documentation, these dimensional features can only

Feature Name	APPROACH_SPEED(KMH)
Parameter	1.4
Level	2
Phase of Declaration	Data Preparation
Generating Algorithm	Pert_Percent
Feature Name	TAIL_HEIGHT(M)
Parameter	2
Level	2
Phase of Declaration	Data Preparation
Generating Algorithm	Pert_Percent
Feature Name	PARKING_AREA(SQM)
Parameter	3
Level	2
Phase of Declaration	Data Preparation
Generating Algorithm	Pert_Perc

Table 4.8: Combined option declarations for features APPROACH_SPEED(KMH), TAIL_HEIGHT(M) and PARKING_AREA(SQM)

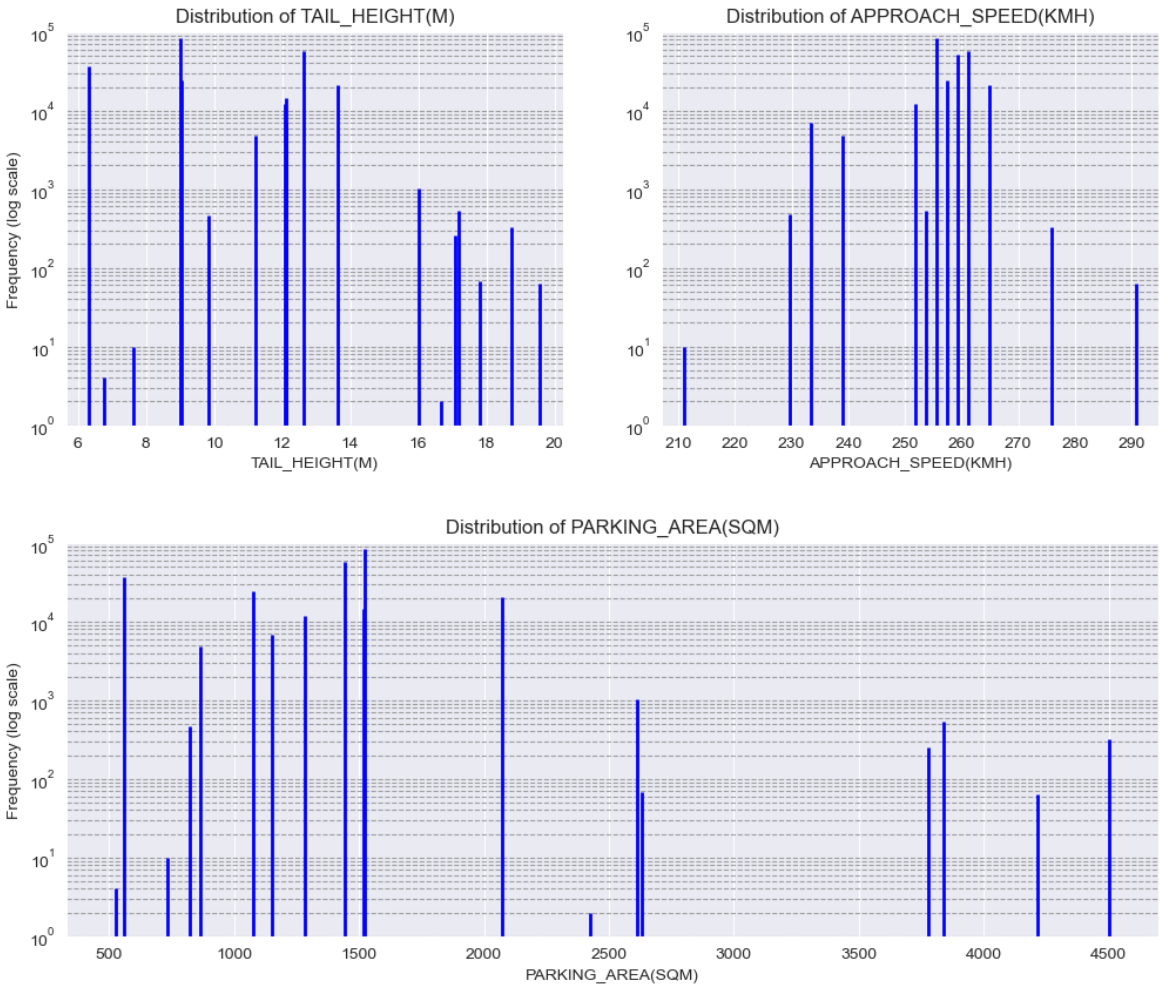


Figure 4.1: Density of aircraft dimension features on logarithmic scale.

hold values that correspond to any of the aircraft types that are realised by constructors. Although some manufacturers develop aircraft models with different specifications in mind, which also result in slightly different dimensions, the number of possible alterations is limited. Furthermore, Chapter 3.3.2 explained that all subtypes of a given aircraft were combined into the most prominent type for each aircraft model during the data preparation phase. Therefore, the variance of the feature was further decreased by eliminating all variations for each aircraft model. Besides this loss of variance within the training data, older and less popular aircraft models, which are oftentimes smaller and lighter, lost representation in the dataset, opening gaps in the training features, indicative in Figure 4.1. Some areas with closely neighbouring unique values are shown at tail heights of around $9m$ and $17m$, parking areas of around $2600m^2$ as well as with the noteworthy cluster at approach speeds of $250km/h$ to $265km/h$. Additionally, Figure 4.1 also indicates many sparse areas with no representation in the dataset. If one were now to alter a given data point by a fixed absolute value or percentage, a trained machine learning model would theoretically react more strongly to a data point that is found in a denser area than it would otherwise. These structural characteristics of the dimensional features aside, the features are also fit for perturbation because a slight change in some aspect of an aircraft's dimensions should not affect the aircraft's tendency to arrive on-time, early, or late. Since there is no general rule by which we can assume a tolerance for a given dimensional spec of an aircraft, as we had for sensor tolerances and similar option declarations, a different approach for finding some margins fit for perturbation is needed, that

is connected to the aforementioned characteristics of the dimensional features. Since there are gaps between the model's dimensional attributes, as seen in Figure 4.1, it was chosen to come up with a metric that tries to perturb input values within these gaps. Furthermore, it is assured that the chosen parameter is based on all data points, including the ones in the sparser area of the training data. Therefore, we looked into the relative distance to the closest value cluster for each cluster and calculated the median over all closest distances to counteract outliers. This median representation is then divided by 2 to represent avoiding overlap between neighbouring value clusters. Following this approach yielded the perturbation margins in relative values of (i) 0.7%, (ii) 2% and (iii) 2.9% for the features (i) *'APPROACH_SPEED(KMH)'*, (ii) *'TAIL_HEIGHT(M)'* and (iii) *'PARKING_AREA(SQM)'* respectively. Therefore, perturbation options for these features are defined as shown in Table 4.8.

Level 1 Options

Level 1 perturbation options are the last possible perturbation option category. Within this thesis, all features found in the training dataset were covered by a perturbation option. Therefore, all features whose perturbation option has not yet been described in the previous sections were covered by level 1 perturbation options. Since level 1 options are not further investigated in Section 4.3, the features and corresponding option definitions are only described vaguely in this paragraph. Features in this category were chosen as such, as (i) there were no defined or readable uncertainty values discovered during data understanding and preprocessing phases, (ii) they had limited unique entries such that perturbations would impact the feature too strongly, or (iii) were too important for the classifier, such that small value changes would expect a change of classification, which goes against the expected goal of the perturbation approach. For example, categorical or ordinal scale features like *'WINGLETS(YN)'* and the weather event features with ranges from 0-3 were covered by level 1 options due to only having two to three unique values. A change between these values demonstrates too drastic of a change to make sense for the perturbation approach. Furthermore, features with high predictive quality like *'DEP_DELAY(MIN)'* are described by level 1 perturbation options, since small margins affect the outcome of the classification task significantly. For important features such as *'DEP_DELAY(MIN)'*, where small changes can typically influence the predictive outcome, the absence of a change of prediction might be just as interesting. Although such investigations are a possibility for features covered by level 1 perturbation options, this thesis does not explore such scenarios.

4.2 Execution of Perturbation

The perturbation approach tries to look into single predictive scenarios with the goal of evaluating their reliability. For this thesis, it was chosen to evaluate the performance of the perturbation approach and its capabilities over a wide variety of data. To allow for this wider analysis, the whole test set of the evaluation process was used to perform perturbation analysis. Therefore, all features of all data vectors in the test set are altered according to the corresponding perturbation option, resulting in 8 new data vectors for every data entry and feature, since all declared options with levels 2 or higher produce 8 perturbed values. This introduces the problem of time complexity and storage necessity to the problem at hand. For the original test dataset of 65.801 entries, the full perturbation run results in $6.316.896$ data vectors that are fed into the classification

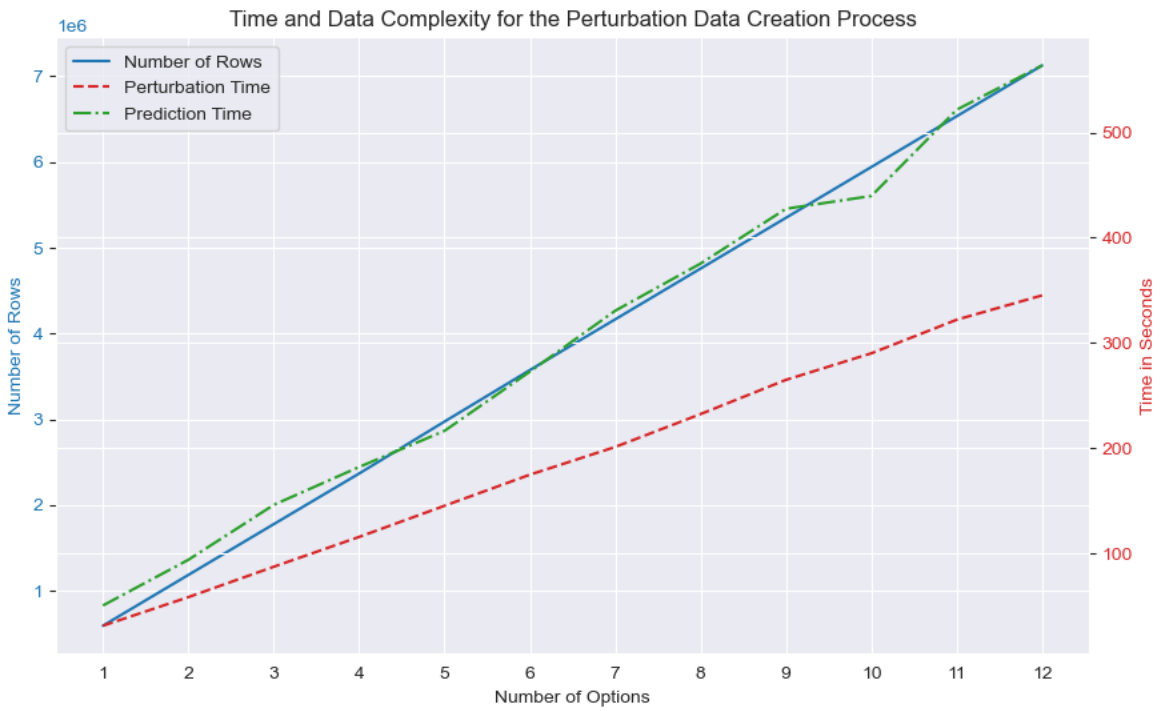


Figure 4.2: Time and data complexity for the perturbation data generation and prediction processes.

model. Notably, this only takes perturbation options with levels ≥ 2 into account. Using more options, combinations of features, or lower-level thresholds would further increase dataset bloat, which in turn increases time complexity and data consumption for the prediction step of the perturbation process. Therefore, it was chosen to only evaluate single-column perturbations to keep this complexity scaling linear, as shown in Figure 4.2. Both the time needed for the perturbation phase and the prediction phase grow linearly with the size of the created dataset. For multi-column perturbation, this complexity would increase exponentially, making a wide-scale analysis not feasible within the scope of this thesis. Because of that, future sections focus solely on single-column perturbations as described. Within the perturbation process, not only are additional rows inserted into the dataset, but also indicators to allow identification for later evaluation processes. These additional columns are appended to the input vector of the test data set on which we perform perturbation analysis. This is needed to evaluate prediction changes, making it possible to backtrack to which input row was the original one.

Table 4.9 shows the structure of the appended columns. *'y_true'* holds the true target value as extracted from the data source. For this thesis, the possible target classes are either 1 (early-arrival), 2 (on-time arrival) and 3 (late arrival). *'pert_id'* informs about the original data row and column of the test set that was responsible for the created perturbation row by saving it in the format of '$column\_name<index>$'. Due to this structure, it is possible to later identify the original data row by splitting at the '<' character. *'pert_level'* shows the level as defined in the corresponding perturbation option, making it possible to later filter for option thresholds. Similar to target classes,

Feature 1	**...**	**Feature N**	**y_true**	**pert_id**	**pert_level**	**y**
value1	...	valueN	{1, 2, 3}	col<original_index>	{1, 2, 3}	{1, 2, 3}

Table 4.9: Example table with attached identification fields.

the perturbation options are represented by the classes 1 (green-level option), 2 (orange-level option) and 3 (red-level option). The last column *'y'* shows the target value as predicted by the machine learning model that was used for the particular run.

4.3 Evaluation

The previously described perturbation processes described the definition of perturbation options, the algorithm for perturbation case creation and the resulting dataset. This section describes the analysis and corresponding insights that are drawn from the created perturbed dataset in detail and points out possible advantages operators might gain by using the perturbation approach within the scopes of this use case. Generally, this section is divided into two parts, namely graphical analysis and quantitative analysis of the generated dataset. For graphical analysis of the perturbation approach, two histograms are created for each feature that is included in a given perturbation run. The first histogram shows the distribution of all entries used as input for the perturbation approach. In this specific case, this input equals the test dataset as described earlier. The second histogram shows the distribution of all entries that lead to an unreliable prediction according to the perturbation approach. Such entries that lead to unreliable predictions are further mentioned as *'alarmed'*. By comparing the original distribution to the *'alarmed'* distribution, areas that are responding stronger or weaker to perturbation are observable. It is assumed that these areas indicate regions in the ML model's predictive space that are more or less sensitive to small changes. This provides the user with valuable insights into the ML model without needing information about the technical characteristics. Furthermore, the *'alarmed'* distributions regarding the varying ML models used in this thesis can be compared to each other to expose differences in the predictive space between the models themselves. Quantitative analysis of the perturbation approach looks into general machine learning metrics like accuracy, precision and recall and tries to give objective measures for the effectiveness of the perturbation approach. The quantitative analysis also aims to prove that the perturbation approach is applicable for real-world datasets and provides benefits when looking into subsamples of the dataset. The predicted dataset is therefore filtered to create two subsamples. One consists of all entries that lead to an unreliable prediction (alarmed), the other consists of the complement that is freed from all unreliable predictions. The performance scores depending on each subset can be compared, showing the potential gain to a system by identifying unreliable cases through perturbation.

Graphical Analysis of Perturbation Cases

The final perturbation dataset provides a data structure as seen in Table 4.9. This structure allows an identification of alarmed and unnoticed data rows according to the perturbation process by grouping with *'pert_id'* and checking if any of the perturbed values within one group are different from the original value of the data row. If such an observation is made, the data row with its corresponding *'pert_id'* is deemed *'alarmed'*. With this distinction, the differences between distributions of the original dataset and all perturbed data can be visualized according to the following graphs. These graphs are further called *'Perturbation Graphs'* and visualize the differences in distribution of the original data's distribution and the distribution of the same column for the alarmed data rows. With this visual comparison, it is assumed that irregularities between the full

test data space in contrast to the *alarmed* space are indicators for areas in which the machine learning algorithm is more sensitive to changes.

Generally, all perturbation graphs depict four separate figures. The top-left shows the distribution of respective values of a given feature, colour coded by the corresponding predicted target variable. The other graphs show the same distribution filtered such that only entries are shown in which perturbation led to differing predictions of the target variable. The top-right graph shows the alarmed distribution of the Random Forest Classifier's (RF) results, the bottom-left graph shows the alarmed distribution created by the Adaptive Boosting Classifier (ADAB) and the bottom-right graph depicts the alarmed distribution of the Gradient Boosting Classifier (XGB). By looking at differences between these distributions, we could observe (i) more sensitive areas for a given predictor, (ii) underrepresented areas for a given predictor and (iii) no outstanding differences between the distributions' shapes. Since we also evaluate the performance of multiple predictors, we can also observe differences between perturbation outcomes of said predictors and observe differences of sensitivities in each of the models. A more sensitive model would show a higher amount of alarmed rows, which is shown by the counter variable n that is shown in the header of each subgraph. Furthermore, the y-axis shows the number of entries identified for each value, which also provides insight into the sensitivity for each specific value.

Figure 4.3: DEWPOINT_TEMP(C) perturbation graph.

Figure 4.3 shows the distribution of alarmed entries for the feature capturing dew point temperature. This feature serves as an excellent example of a perturbation case that is significantly more sensitive at higher values. All evaluated models hardly reacted to perturbation for values lower than $10°C$. Furthermore, it is observable that all models have similar peaks for the most occurring bin at around 100 to 150 alarmed entries. However, the random forest model is more sensitive at temperatures around $16°C$, while the XGB model shows to be more sensitive to changes for temperatures around $23°C$. Besides these observations, the ADAB model is less sensitive overall but also more selective. The highest magnitude of alerts also occurs around the $16°C$ mark similar to the random forest model, but neighbouring temperature values are not affected as strongly, resulting in these narrow clusters of perturbation alerts.

Sea Level Pressure (Millibar) is a feature which is also observed by a level 3 perturbation option. Looking at the perturbation graphs seen in Figure 4.4 shows that the distribution of alarmed cases

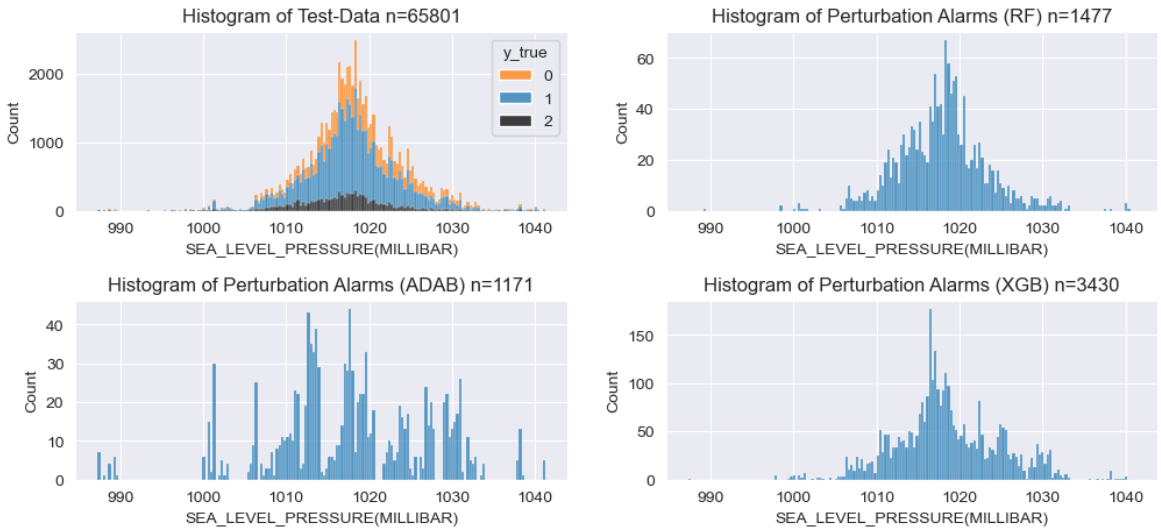


Figure 4.4: SEA_LEVEL_PRESSURE(MILLIBAR) perturbation graph.

for the random forest and XGB models roughly follows the distribution of the underlying test data set. Notably, the shape of the distribution regarding alarmed cases for the ADAB model does not fit the original distribution at all. Furthermore, the ADAB model is also over proportionately sensible on values on the lower and higher extremes of the scale. Therefore, this feature gives an excellent example of perturbation options acting strongly different for varying machine learning processes.

Figures 4.5 and 4.6 show two of the level 2 perturbations handling dimensions of carriers. The latter of which has the unique characteristic that all three models have distinct distributions when it comes to their respective perturbation graphs. The ADAB model does not show strong sensitivity, as outlined by the peak of 20 alarmed entries with a total of 24 alarmed cases. However, contrary to the other two classifiers, the peak is not located on the majority bin. Instead, the most frequent data value is shown to be at around 800 square meters. It is noteworthy that the neighbourhood around this mark does not show high density. Both the random forest and XGB models have the highest peak at 1500 square meters, which is occupied by the majority of carriers in the dataset.

Figure 4.5: APPROACH_SPEED(KMH) perturbation graph.

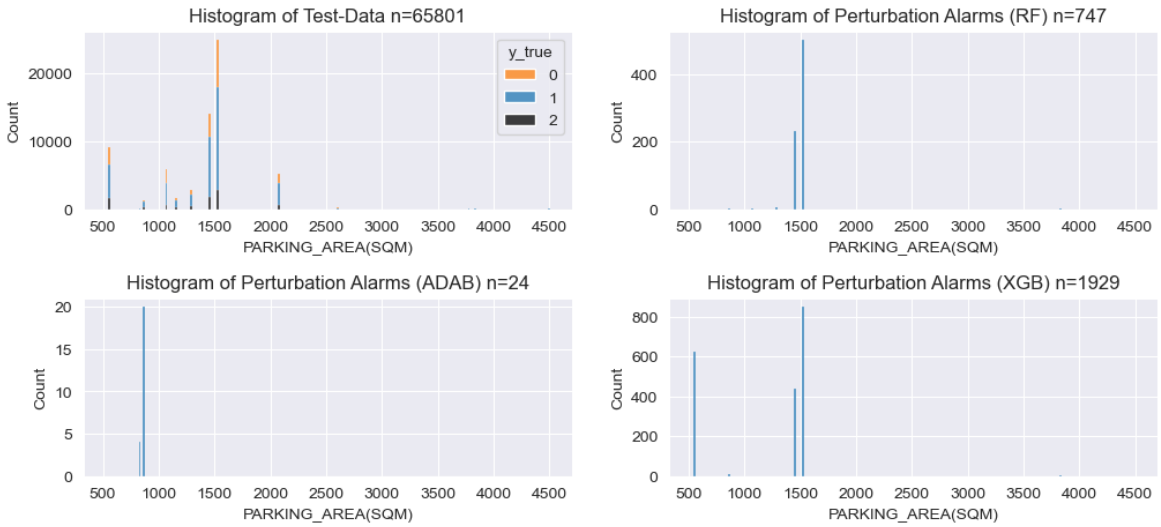


Figure 4.6: PARKING_AREA(SQM) perturbation graph.

Differing results between the random forest and XGB models are found on the lower end of the spectrum, where the random forest does not show any alarmed values while XGB is sensitive to deviations for smaller aircraft types. The 'APPROACH_SPEED(KMH)' feature does not show these deviations, although the perturbation logic is similar and the features correlate strongly to each other.

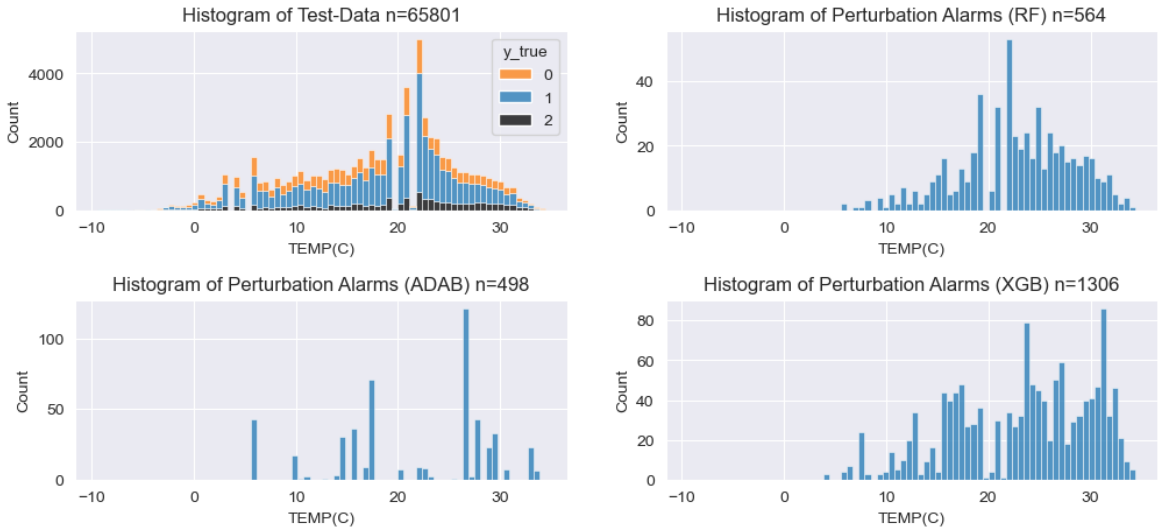


Figure 4.7: TEMP(C) perturbation graph

A further case with varying results for perturbation graphs for each model is the feature 'TEMP(C)' depicted in Figure 4.7. The trend of adaptive boosting being less sensitive retains. However, in contrast to other graphical comparisons, random forest and gradient boosting share similar over-all sensitivity, with slightly different focus points being evident by differences for their respective peak values. Moreover, the XGB model produced slightly more unreliable predictions on the $16°C$-$20°C$ range. Although this cluster of unreliable predictions does not deviate significantly, the difference is evident and could be interesting for further analysis.

Lastly, Figure 4.8 shows the perturbation graph for the feature capturing relative humidity, which is a unique feature due to its characteristic of its perturbation option parameter being dependent on two sensor values. Even though this feature holds a wide perturbation margin with a deviation

of up to 5.445%, the overall number of alerted entries is not particularly outstanding, nor do the shapes of the distributions between the model's respective perturbation graphs differ significantly. The feature follows the same trends, with ADAB generally being less sensitive to changes and XGB being the most sensitive. Furthermore, the random forest model is the only model that has a more pronounced cluster at the 50% relative humidity mark than what is shown by the test dataset's distribution, seemingly having stronger sensitivities in certain regions of the dataset for this feature. For the previously evaluated perturbation graphs, this sensitivity to certain data ranges was observed to be a characteristic of the XGB model, making this particular feature a standout in that regard.

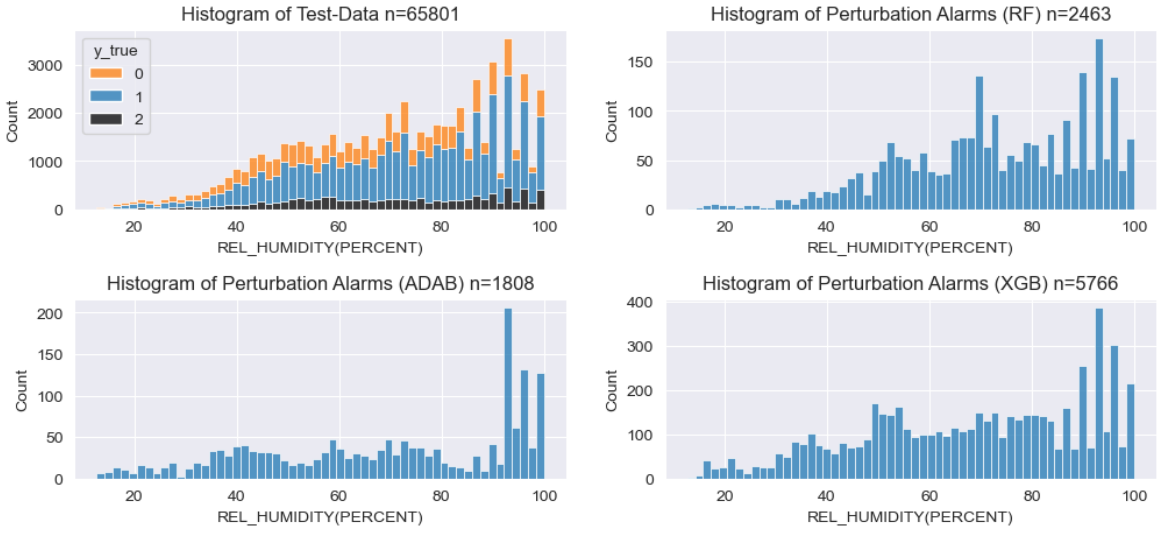


Figure 4.8: REL_HUMIDITY(PERCENT) perturbation graph

With the graphical analysis, only the results from 6 of the 12 perturbation options were shown. All other cases are either behaving similarly or do not show interesting deviations from the test dataset's distribution. The overall trends regarding the behaviour for each described ML model kept consistent throughout all perturbation results. Generally, the graphical analysis of perturbation shows that the random forest classifier behaves similarly to the gradient boosting classifier for nearly all features, indicating similar boundaries and sensitivities. The gradient boosting classifier, in contrast to both other models, is more sensitive to small changes of input values overall. Therefore, the XGB model produces more unreliable predictions than the other classifiers. The adaptive boosting classifier is shown to be less sensitive within the scope of this use case, but focuses more strongly on particular areas of the dataset that are not necessarily the most frequent areas in the test dataset. Therefore, it is evident that the graphical analysis of perturbation results is capable of pointing to possible differences between trained machine learning models and showing areas that are more sensitive to the small alterations to input vectors that are being applied during the perturbation process. However, to really understand underlying causes, operators of machine learning solutions have to look deeper into the weights and boundaries of the respective model. For fast analysis that does not require in-depth insight into the development stages of a model, graphical perturbation analysis might prove useful to boost model understanding and explain differences between trained models at quick glances. Furthermore, operators are made aware of sensitive areas within the respective model's predictive space. This might lead operators to further investigate predictive outcomes that relied on inputs lying within the more sensitive areas if resources for more profound analysis are available.

Quantitative Analytics of Perturbation Cases

Previously described graphical analysis is suitable to find patterns and areas of high sensitivity of the machine learning models on a case-by-case basis. In order to further understand the possible benefits perturbation provides for this use case, the effects of perturbation on traditional machine learning metrics are observed. Therefore, the *'alarmed'* flag is used on the test data set on which the perturbation process is performed on in order to create subsets of the data with $alarmed == true$ and $alarmed == false$. Then, the differences in traditional machine learning metrics between the original data and the created subset are evaluated. Lastly, the maximum possible positive effects an operator of a predictive system is capable of producing by looking into perturbed cases and manually fixing the system are looked into. By showing that the ML metric scores are higher on the subsets that are devoid of unreliable predictions, one can assume that perturbation was successful in identifying actual unreliable predictive scenarios. Contrary, if the subset would show equal metric scores regardless of the inclusion of unreliable predictions, the perturbation approach would fail to provide quantitative benefits for operators of predictive systems.

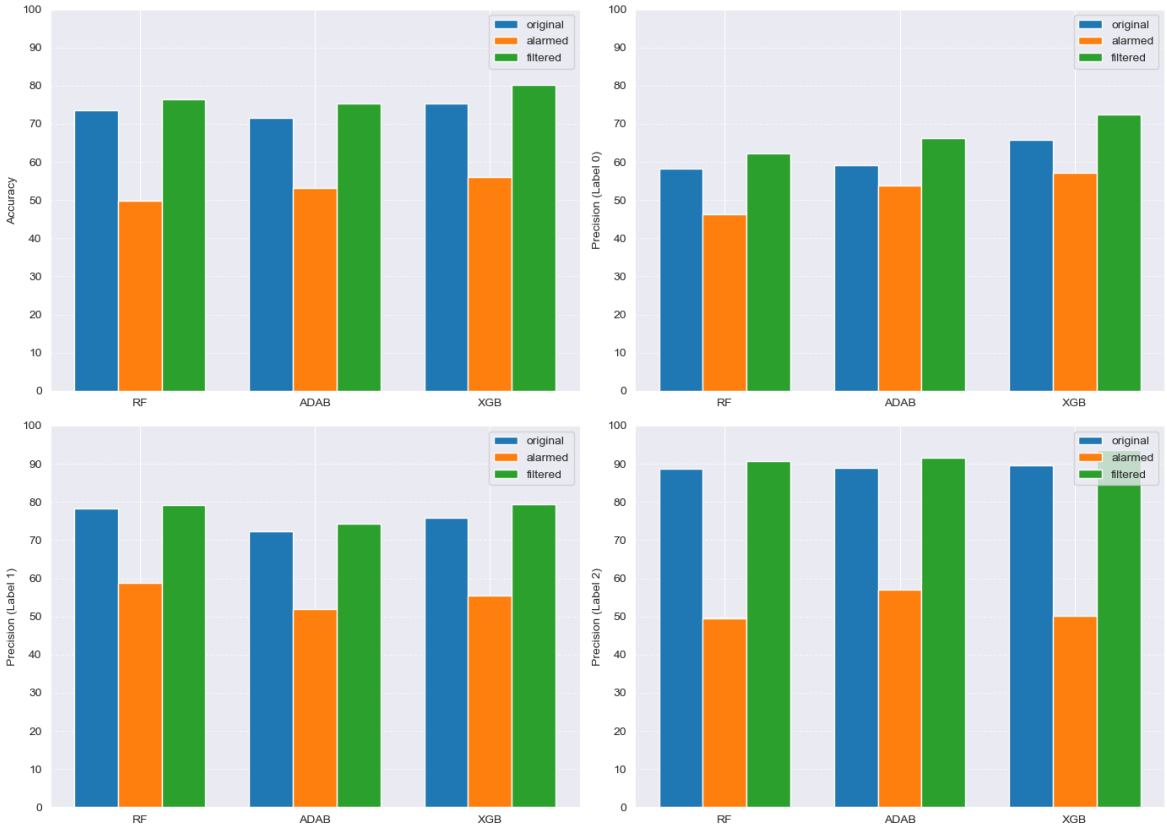


Figure 4.9: Impact of perturbation subset-evaluation on evaluation metrics.

By separating the dataset based on perturbation alarms, the subsets are comparable in terms of their evaluation metrics. For this thesis, we look into accuracy and precision for each target class in order to get a broad look into the performance of the models on each subset. Figure 4.9 shows the performance metrics for all models used in the scope of this thesis. Each quadrant in Figure 4.9 shows the performance of each model and output subset combination for either (i) accuracy, (ii) precision regarding early arrivals (label 0), (iii) precision regarding on-time arrivals (label 1) and precision regarding late arrivals (label 2). The blue bar represents the corresponding metric

that has been achieved over all test entries. The orange bar shows the metrics when limiting the evaluated test set to only include unreliable predictions (alarmed). Finally, the green bar indicates the corresponding metric on a test set that is devoid of all unreliable predictions (filtered). Based on this experimental analysis, the possible benefits of perturbation are observable. For all metrics, the subset of alarmed cases leads to a drastic drop in metric scores. This is an indicator that perturbation is capable of identifying less accurate areas for the machine learning classifiers by identifying sensitive areas through simply altering input vectors within small margins. In contrast, all metrics slightly improve for non-alarmed cases. The extent of the improvement is smaller, since the total number of alarmed entries is generally low. However, the positive impact is observable and therefore not insignificant.

This positive impact of identifying highly sensitive areas within a model's data space can be further leveraged by operators of predictive systems, which often have prior experiences within their respective domains. With this knowledge of internal processes and patterns, they are potentially capable of interpreting edge cases better when it comes to unreliable predictions made automatically by ML models. The perturbation process, as outlined in this thesis, is capable of finding such unreliable predictions for flight delay systems, which allows operators to specifically investigate the alarmed cases as found out by the perturbation process as previously explained. In optimal settings, an observer could, in theory, look into all alarmed cases and change their prediction to the correct outcome. Although highly unlikely, this gives an upper limit to the possible benefit of the perturbation approach in a quantitative sense. On the flip side, one can observe the maximum possible damage an observer could create by misclassifying all alarmed cases that were rightfully recognized by the machine learning model. This provides users with the understanding of the lower bound to evaluate the perturbation approach and the applicability of the perturbation approach for the predictive task on which it is used to check individual prediction's reliability.

Table 4.10 shows all final percentage values for accuracy and micro precision values for all possible classes. Precisions are denoted with an identifier that is resolved according to the logic of: 0 = *early; 1=on_time; 2=late*. Table 4.10 also shows that for all classifiers, the alarmed test dataset, which contains only unreliable predictions, yields drastically worse performance metrics for all evaluated metrics compared to the full test dataset. Since the complementary subset then only consists of reliable predictions, it shows that all evaluated performance metrics scored higher compared to the original dataset, as evident in the *'No Alarmed'* rows. The extent of the positive change is smaller than the negative performance influence on the alarmed subset. The smaller positive influence on performance metrics can be explained by the alarmed subset's significantly smaller size. It consists of 11.02% to 20.24% of data entries compared to the original dataset, depending on the model used for evaluation. With this smaller size, the elimination of the alarmed entries cannot lead to drastic changes of metric scores.

The Random Forest Classifier (RF) identifies the fewest unreliable test cases with $p_a = 11.02\%$, while the Gradient Boosting Classifier (XGB) contained the highest number of unreliable test cases with $p_a = 20.24\%$. Looking into the specific metric scores, Table 4.10 shows that, with an accuracy score of 49.90%, the Random Forest Classifier has the worst performing alarmed subset. On the alarmed subset, the Gradient Boosting Classifier and the Adaptive Boosting Classifier (ADAB) reached accuracy scores of 55.95% and 53.22% respectively. This indicates that the perturbation approach, when used with the RF, was more likely to identify unreliable predictions

Random Forest Classifier ($p_a = 7249/65801$)				
Category	**Accuracy**	**Precision 0**	**Precision 1**	**Precision 2**
Real Test Results	73.43%	58.30%	78.22%	88.78%
Only Alarmed	49.90%	46.27%	58.79%	49.56%
No Alarmed	76.34%	62.31%	79.27%	90.77%
All Fixed	78.95%	68.68%	81.25%	91.24%
Gradient Boosting Classifier ($p_a = 13315/65801$)				
Category	**Accuracy**	**Precision 0**	**Precision 1**	**Precision 2**
Real Test Results	75.19%	65.86%	75.88%	89.50%
Only Alarmed	55.95%	57.20%	55.41%	50.08%
No Alarmed	80.07%	72.32%	79.46%	93.59%
All Fixed	84.10%	84.06%	82.45%	94.38%
Adaptive Boosting Classifier ($p_a = 11600/65801$)				
Category	**Accuracy**	**Precision 0**	**Precision 1**	**Precision 2**
Real Test Results	71.46%	59.07%	72.30%	89.01%
Only Alarmed	53.22%	53.69%	51.88%	56.97%
No Alarmed	75.37%	66.22%	74.19%	91.59%
All Fixed	79.71%	83.10%	77.11%	92.51%

Table 4.10: Evaluation metrics for all models/baselines depending on perturbation subsets. (Level >= 2)

which turned out to be misclassified by the system anyway. Additionally, by simply discarding all predictions marked as unreliable from the system, accuracy scores of 76.34% (RF), 80.03% (XGB) and 75.37% (ADAB) are shown for the models. This translates to an increase in accuracy scores of 2.91%, 4.88% and 3.91% respectively. Here, the Gradient Boosting Classifier shows to benefit the strongest when it comes to curating the model's output by discard unreliable predictions. This advantage is further leveraged by assuming a perfect operator's potential of manually fixing all unreliable predictions to the true target class. With this assumption, the accuracy scores in the predictive system's curated output would show to be 78.95% (RF), 84.10% (XGB) and 79.71% (ADAB). This procedure would increase the scores compared to the original output by 5.52%, 8.91% and 8.25% respectively. With the manually curated system output, the Adaptive Boosting Classifiers achieved accuracy scores would theoretically surpass the Random Forest Classifier's achieved scores. However, this stronger increase in scores is a result of the higher number of alarmed cases, which would result in needing more manual checking of the system's output. The precision scores for the corresponding target classes follow the same pattern as described for the accuracy scores and are therefore not further explored.

Generally, the Random Forest Classifier showed to produce the least amount of unreliable predictions. Furthermore, the predictions that have been marked as unreliable are more likely to have been actually misclassified by the predictor. However, by discarding unreliable predictions and therefore automatically curating the systems' output to only include reliable predictions, the Gradient

Boosting Classifier further outperforms the Random Forest Classifier due to its already better accuracy without curating the output. With manually correcting unreliable predictions, the performances of the final output can be improved additionally. However, operators would have to manually check 11.02% of the Random Forest Classifier's output and up to 20.24% of the Gradient Boosting Classifier's output. The feasibility for manual checking in such grand scopes needs to be evaluated by the operators on a case-by-case basis. In conclusion, the perturbation approach worked best at identifying actual problematic predictions in combination with the Random Forest Classifier, while the Gradient Boosting Classifier's output could be improved the most due to the automatic or manual curation of the predictive system's output.

5

Summary

This thesis evaluates the usability of the reliability checking process with respect to a flight delay classification problem within the aviation industry. Following CRISP-DM phases with the additional required documentation steps, as defined by the reliability checking process, lead to the development of three classifiers, namely (i) Random Forest Classifier, (ii) Gradient Boosting Classifier, (iii) Adaptive Boosting Classifier. These classifiers were trained on flight scheduling data, meteorological air reports, aircraft characteristics and notice to airmen messages and were capable of predicting domestic flights arriving at Hartsfield-Jackson Airport in Atlanta (ATL) with a test-accuracy of 73.43%, 75.15% and 71.46% respectively, within a multi-class classification task identifying early, on-time and late arrivals at ATL. To assess the reliability of individual flight predictions, the perturbation approach was adapted.

This thesis contributes to the reliability checking process by evaluating the results of the perturbation approach used for reliability assessment in a real-world scenario for domestic flight arrival delay classification. The three classifiers built for this thesis provided a baseline on which the final evaluation phases of the perturbation approach were performed on. Splitting model predictions into subsets based on the occurrence of perturbation alarms allows for a model agnostic analysis of sensitive areas of a given machine learning model. The output of the perturbation approach is evaluated either (i) graphically, by comparing distributions of unreliable cases to the original dataset's distribution, or (ii) quantitatively, by looking into the impact on performance metrics after eliminating or correcting unreliable predictions from the predicted dataset. Graphical analysis can assist in finding differences between boundaries of models trained on the same data set. Quantitative analysis, as done in this thesis, allows the user to identify individual predictions leading to worse accuracy and macro precision by looking into traditional evaluation metric differences for each subset, depending on including or discarding predictions marked as unreliable. For the three classifiers of Random Forest Classifier, Gradient Boosting Classifier and Adaptive Boosting Classifier that have been adopted within the scope of this thesis, the accuracies of the test sets were risen from 73.43% to 76.34%, from 75.19% to 80.07% and from 71.46% to 75.32% respectively, by eliminating the subset of unreliable data entries from the evaluation. Experienced operators or more specialized machine learning models could theoretically correct the incorrectly classified predictions that also have been marked as unreliable within the perturbation approach. The associated improvement to the predictive output would change the accuracy scores to 78.95%, 84.10% and 79.71% respective to their corresponding models, following the assumption of a perfect guesser. Although highly unlikely, it provides an upper bound of what is possible within this real-life

use case when it comes to improving quantitative metrics of the machine learning process by using the perturbation approach. For the detection of unreliable predictions, the perturbation process based on the Random Forest Classifier identified the lowest number with 11.02% of all test cases leading to unreliable predictions according to the perturbation approach. The Gradient Boosting Classifier identified 20.24% of unreliable predictions. Therefore, the Random Forest Classifier showed to be less sensitive to small alterations of input fields, which increases the trustworthiness of the predictor for operators compared to the other evaluated models. Additionally, a smaller number of unreliable predictions is more manageable when it comes to manually improving the system's predictions through domain knowledge. It is also shown that the unreliable predictions identified by perturbation with the Random Forest Classifier are more likely to include misclassifications than other models. However, the Gradient Boosting Classifier reached the highest accuracy score and also benefits from perturbation by having a higher accuracy for predictions that have been identified as reliable. Generally, the perturbation approach was capable of identifying sensitive areas within the model's feature space and lead to higher accuracy scores for predictions that have been found to be reliable.

5.1 Model Limitations

The three models used within this thesis are all based on Decision Tree Classifiers. Random Forest Classifier is an ensemble method based on bagging, while the Gradient Boosting Classifier and Adaptive Boosting Classifier use boosting to enhance the models' performance. These models were chosen due to their speed and simplicity of usage in addition to their fast adaptability to the data used for this thesis, which other simple classifiers were unable to pick up on. Due to this model selection and the simplicity of the model building stage, the performance of the models themselves are not particularly performative in comparison to machine learning models used in the field as shown in other works mentioned in Chapter 2.3. Although the thesis' focus lies on identifying volatilities within the feature space for the collected aviation-specific data as well as using these volatilities to identify unreliable predictions in a model agnostic way, the models' performances and characteristics might influence the capabilities to identify unreliable predictions.

5.2 Process Benefits and Limitations

By using the reliability assessment process, key challenges of machine learning projects are tackled. First, looking into all phases of CRISP-DM with additional focus on documentation and data gathering, forces the use of best practices by data scientists regarding documentation practices. This allows for a deeper understanding of machine learning projects and better interoperability between development teams and operators. Additionally, the perturbation approach tries to explain physical limits within the data understanding phase, like sensor tolerances and data sparsity. Besides better documentation of model building processes, the perturbation approach allows for improved understanding of edge cases produced by classification systems without needing knowledge about the model's characteristics. Instead, perturbing sensitive attributes within defined margins showed to provide insights into the predictive space of the model in a model agnostic way. Furthermore, using the perturbation approach for reliability assessment enables operators of ML system to be capable of evaluating the reliability of individual predictions without needing knowledge of the

used model itself. Automatic detection of edge cases could then be investigated in more detail to alleviate lacklustre model performances in local areas. Especially for domains with high criticality, where wrongful predictions lead to high costs or even endanger life, identifying the reliability of individual predictions is critical. The perturbation approach within the reliability assessment process is a rather simple adaptation for reliability assessment of individual predictions and might be especially useful to domains in need of reliable predictive systems.
Although the proposed process showed to work within the scope of this thesis regarding the aviation domain, in other projects the feature space might not include data with inherent vagueness due to sensor tolerances or other limiting factors. Furthermore, the quantitative benefits of the perturbation approach, as described in this thesis, are not generalizable to other data mining projects. Since the process is bound to specific use-cases, each use-case might respond differently to the quantitative and visual analysis described. Therefore, more studies covering various domains are needed to evaluate the generalizability of the approach. Lastly, perturbation leads to the creation of slightly altered data points on runtime to check reliability. This increases the size of the dataset significantly. With increased amount of data, time complexity increases depending on the model used. Within the scope of this thesis, only single column perturbations were considered, since multi-column analysis over all test data was not feasible with the data's exponential increase. Although resource intensive, perturbation might be a worthwhile endeavour for industries relying on precise predictive systems nonetheless.

5.3 Future Work

The reliability checking process, as defined during the scope of the flight delay prediction data mining project, proved to be capable of identifying unreliable predictions for this specific use case. Furthermore, it showed quantitatively to improve overall model performance by a significant margin by correcting or discarding unreliable predictions. To arrive at a more generalisable assertion, this process should be adopted to more domains and projects covering a wider amount of various data mining processes, patterns and predictive systems. Using the perturbation approach for reliability assessment on a model that is used in production would also improve the validity of the process. Machine learning systems deployed in productive systems need to fulfil stricter quality criteria than the models proposed in this thesis. Therefore, perturbation analysis on a system that has already proven to be effective in the field is necessary to fully grasp the capabilities of the approach.
Manually fixing predicted data within this thesis has proven to improve the model performances for all model types significantly. To build on this insight, perturbation could be used in multi-stage predictive systems, in which the first stage uses a machine learning model intending to solve a problem as it would in a single-stage project. The second stage could then build up on identified patterns, correcting shortcomings of the initial model by using the knowledge gained through perturbing input vectors.
Lastly, the proposed reliability assessment approach only considers classification problems. Extending the scope of the process into regression tasks would allow covering a wider scope of data mining projects to evaluate the perturbation approach. Furthermore, exploring ways to analyse multi-column perturbation in addition to single-column perturbation is needed to expand the meaningfulness for the adoption of the perturbation process.

Bibliography

[1] Buerau of Transportation Statistics. '2019 traffic data for u.s. airlines and foreign airlines u.s. flights - final, full-year.' (6th Nov. 2020), [Online]. Available: https://www.bts.dot.gov/newsroom/final-full-year-2019-traffic-data-us-airlines-and-foreign-airlines (visited on 15/11/2023).

[2] T. Jain, V. Menon, D. Nease, C. Robins and K. Sattary, 'An analysis of COVID-19's impact on u.s. aviation,' 1st Dec. 2020, p. 11 851.

[3] F. R. B. of St. Louis. 'Enplanements for u.s. air carrier domestic and international, scheduled passenger flights.' (25th Aug. 2025), [Online]. Available: https://fred.stlouisfed.org/series/ENPLANE (visited on 20/09/2025).

[4] M. Ball, C. Barnhart, M. Dresner *et al.*, 'Total delay impact study : A comprehensive assessment of the costs and impacts of flight delay in the united states,' 1st Oct. 2010. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/6234 (visited on 07/11/2023).

[5] S. Kapoor and A. Narayanan, 'Leakage and the reproducibility crisis in machine-learning-based science,' *Patterns*, vol. 4, no. 9, p. 100 804, 8th Sep. 2023, ISSN: 2666-3899. DOI: 10.1016/j.patter.2023.100804.

[6] S. Staudinger, C. G. Schuetz and M. Schrefl, 'A reference process for assessing the reliability of predictive analytics results,' *SN Computer Science*, vol. 5, no. 5, p. 563, 18th May 2024, ISSN: 2661-8907. DOI: 10.1007/s42979-024-02892-4.

[7] X. Zhang and I. Bose, 'Reliability estimation for individual predictions in machine learning systems: A model reliability-based approach,' *Decision Support Systems*, vol. 186, p. 114 305, 1st Nov. 2024, ISSN: 0167-9236. DOI: 10.1016/j.dss.2024.114305.

[8] R. Wirth and J. Hipp, 'CRISP-DM: Towards a standard process model for data mining,' *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, Jan. 2000.

[9] V. Plotnikova, M. Dumas and F. P. Milani, 'Applying the CRISP-DM data mining process in the financial services industry: Elicitation of adaptation requirements,' *Data & Knowledge Engineering*, vol. 139, p. 102 013, 2022, ISSN: 0169-023X. DOI: 10.1016/j.datak.2022.102013.

[10] V. Plotnikova, M. Dumas and F. Milani, 'Adaptations of data mining methodologies: A systematic literature review,' *PeerJ Computer Science*, vol. 6, e267, 25th May 2020, ISSN: 2376-5992. DOI: 10.7717/peerj-cs.267.

[11] M. Bardach, E. Gringinger, M. Schrefl and C. G. Schuetz, 'Predicting flight delay risk using a random forest classifier based on air traffic scenarios and environmental conditions,' in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, Oct. 2020, pp. 1–8. DOI: 10.1109/DASC50938.2020.9256474.

[12] B. Thiagarajan, L. Srinivasan, A. V. Sharma, D. Sreekanthan and V. Vijayaraghavan, 'A machine learning approach for prediction of on-time performance of flights,' in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, ISSN: 2155-7209, Sep. 2017, pp. 1–6. DOI: 10.1109/DASC.2017.8102138.

[13] M.-T. Vo, T.-V. Tran, D.-T. Pham and T.-H. Do, 'A practical real-time flight delay prediction system using big data technology,' in *2022 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, Nov. 2022, pp. 160–167. DOI: 10.1109/COMNETSAT56033.2022.9994427.

[14] Y. Jiang, Y. Liu, D. Liu and H. Song, 'Applying machine learning to aviation big data for flight delay prediction,' in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, Aug. 2020, pp. 665–672. DOI: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00114.

[15] R. Nigam and K. Govinda, 'Cloud based flight delay prediction using logistic regression,' in *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, Dec. 2017, pp. 662–667. DOI: 10.1109/ISS1.2017.8389254.

[16] R. Hendrickx, M. Zoutendijk, M. Mitici and J. Schäfer, 'Considering airport planners' preferences and imbalanced datasets when predicting flight delays and cancellations,' in *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, ISSN: 2155-7209, Oct. 2021, pp. 1–10. DOI: 10.1109/DASC52595.2021.9594367.

[17] G. Gui, F. Liu, J. Sun, J. Yang, Z. Zhou and D. Zhao, 'Flight delay prediction based on aviation big data and machine learning,' *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 140–150, Jan. 2020, Conference Name: IEEE Transactions on Vehicular Technology, ISSN: 1939-9359. DOI: 10.1109/TVT.2019.2954094.

[18] L. Moreira, C. Dantas, L. Oliveira, J. Soares and E. Ogasawara, 'On evaluating data preprocessing methods for machine learning models for flight delays,' in *2018 International Joint Conference on Neural Networks (IJCNN)*, ISSN: 2161-4407, Jul. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489294.

[19] Z. Bosnić and I. Kononenko, 'Comparison of approaches for estimating reliability of individual regression predictions,' *Data & Knowledge Engineering*, vol. 67, no. 3, pp. 504–516, 1st Dec. 2008, ISSN: 0169-023X. DOI: 10.1016/j.datak.2008.08.001.

[20] C. Mayer and T. Sinai, 'Network effects, congestion externalities, and air traffic delays: Or why not all delays are evil,' *American Economic Review*, vol. 93, no. 4, pp. 1194–1215, 2003, ISSN: 0002-8282. DOI: 10.1257/000282803769206269.

[21] Airlines for America. 'U.s. passenger carrier delay costs,' Airlines for America. (17th Nov. 2025), [Online]. Available: https://www.airlines.org/dataset/u-s-passenger-carrier-delay-costs/ (visited on 17/11/2025).

[22] E. Mitsokapas, B. Schäfer, R. J. Harris and C. Beck, 'Statistical characterization of airplane delays,' *Scientific Reports*, vol. 11, no. 1, p. 7855, 12th Apr. 2021, Publisher: Nature Publishing Group, ISSN: 2045-2322. DOI: 10.1038/s41598-021-87279-8.

[23] W. Deng, B. Li and H. Zhao, 'Study on an airport gate reassignment method and its application,' *Symmetry*, vol. 9, no. 11, p. 258, Nov. 2017, Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2073-8994. DOI: 10.3390/sym9110258.

[24] R. Dhanawade, M. Deo, N. Khanna and R. V. Deolekar, 'Analyzing factors influencing flight delay prediction,' in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, Mar. 2019, pp. 1003–1007.

[25] Buerau of Transportation Statistics. 'Fields of on-time : Reporting carrier on-time performance (1987-present),' OST_R | BTS | Transtats Fields. (2020), [Online]. Available: https://transtats.bts.gov/DatabaseInfo.asp?QO_VQ=EFD&DB_URL=Z1qr_VQ=E&Z1qr_Qr5p=N8vn6v10&f7owrp6_VQF=D (visited on 20/09/2023).

[26] Buerau of Transportation Statistics. 'Overview of airline on-time performance data,' OST_R | BTS | Transtats Databases. (2020), [Online]. Available: https://transtats.bts.gov/Fields.asp?gnoyr_VQ=FGJ (visited on 20/09/2023).

[27] Federal Aviation Administraion. 'Aircraft characteristics database | federal aviation administration,' Aircraft Characteristics Database. (1st Jul. 2025), [Online]. Available: https://www.faa.gov/airports/engineering/aircraft_char_database (visited on 14/06/2026).

[28] Iowa State University, Department of Agronomy. 'Iowa environmental mesonet: Data download,' Iowa Environmental Mesonet. (2024), [Online]. Available: https://mesonet.agron.iastate.edu/request/download.phtml (visited on 14/06/2026).

[29] V. L. Nadolski, *Automated surface observing systems (asos) user's guide*, National Oceanic and Atmospheric Administration (NOAA), Mar. 1998. [Online]. Available: https://www.weather.gov/media/asos/aum-toc.pdf (visited on 24/10/2023).

[30] National Weather Service. 'Metar decode key.' (24th Dec. 2008), [Online]. Available: https://www.weather.gov/media/wrh/mesowest/metar_decode_key.pdf (visited on 14/06/2026).

[31] Federal Aviation Administration. 'Notam search.' (2026), [Online]. Available: https://notams.aim.faa.gov/notamSearch/nsapp.html/ (visited on 14/06/2026).

[32] Federal Aviation Administration. 'What is a NOTAM?' (30th Sep. 2025), [Online]. Available: https://www.faa.gov/about/initiatives/notam/what_is_a_notam (visited on 14/06/2026).

[33] G. S. Daş, F. Gzara and T. Stützle, 'A review on airport gate assignment problems: Single versus multi objective approaches,' *Omega*, vol. 92, p. 102 146, 1st Apr. 2020, ISSN: 0305-0483. DOI: 10.1016/j.omega.2019.102146.

[34] D. Truong, 'Using causal machine learning for predicting the risk of flight delays in air transportation,' *Journal of Air Transport Management*, vol. 91, p. 101 993, 1st Mar. 2021, ISSN: 0969-6997. DOI: 10.1016/j.jairtraman.2020.101993.

[35] P. Hu, J. Zhang and N. Li, 'Research on flight delay prediction based on random forest,' in *2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, Oct. 2021, pp. 506–509. DOI: 10.1109/ICCASIT53235.2021.9633476.

[36] K. Zhang, Y. Jiang, D. Liu and H. Song, 'Spatio-temporal data mining for aviation delay prediction,' in *2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC)*, ISSN: 2374-9628, Nov. 2020, pp. 1–7. DOI: 10.1109/IPCCC50635.2020.9391561.

[37] Boeing Commercial Airplanes, *Airplane characteristics for airport planning: Boeing 737*, Sep. 2020. [Online]. Available: https://www.boeing.com/content/dam/boeing/boeingdotcom/commercial/airports/acaps/737_RevA.pdf (visited on 14/06/2026).

[38] Boeing Commercial Airplanes, *Airport compatibility engineering*, 16th Mar. 2016. [Online]. Available: https://www.boeing.com/content/dam/boeing/boeingdotcom/commercial/airports/faqs/arcandapproachspeeds.pdf (visited on 14/06/2026).

[39] D. Keiser, L. H. Schnoor, B. Pupkes and M. Freitag, 'Life cycle assessment in aviation: A systematic literature review of applications, methodological approaches and challenges,' *Journal of Air Transport Management*, vol. 110, p. 102 418, 1st Jul. 2023, ISSN: 0969-6997. DOI: 10.1016/j.jairtraman.2023.102418.

[40] H. Pelletier. 'Cyclical encoding: An alternative to one-hot encoding for time series features,' Towards Data Science. (3rd May 2024), [Online]. Available: https://towardsdatascience.com/cyclical-encoding-an-alternative-to-one-hot-encoding-for-time-series-features-4db46248ebba/ (visited on 25/02/2025).

[41] P. Domingos, 'A few useful things to know about machine learning,' *Commun. ACM*, vol. 55, no. 10, pp. 78–87, 1st Oct. 2012, ISSN: 0001-0782. DOI: 10.1145/2347736.2347755.

[42] K. Jamieson and A. Talwalkar, 'Non-stochastic best arm identification and hyperparameter optimization,' in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, PMLR, 2nd May 2016, pp. 240–248.

[43] L. Breiman, 'Random forests,' *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[44] M. Pal, 'Random forest classifier for remote sensing classification,' *International Journal of Remote Sensing*, vol. 26, no. 1, pp. 217–222, 1st Jan. 2005, ISSN: 0143-1161. DOI: 10.1080/01431160412331269698.

[45] Y. Freund and R. E. Schapire, 'A decision-theoretic generalization of on-line learning and an application to boosting,' *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1st Aug. 1997, ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1504.

[46] T. Hastie, R. Tibshirani and J. Friedman, 'Boosting and additive trees,' in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, T. Hastie, R. Tibshirani and J. Friedman, Eds., New York, NY: Springer, 2009, pp. 337–387, ISBN: 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7_10.

[47] M. Saied, S. Guirguis and M. Madbouly, 'A comparative study of using boosting-based machine learning algorithms for IoT network intrusion detection,' *International Journal of Computational Intelligence Systems*, vol. 16, no. 1, p. 177, 9th Nov. 2023, ISSN: 1875-6883. DOI: 10.1007/s44196-023-00355-x.

[48] A. J. Ferreira and M. A. T. Figueiredo, 'Boosting algorithms: A review of methods, theory, and applications,' in *Ensemble Machine Learning: Methods and Applications*, C. Zhang and Y. Ma, Eds., New York, NY: Springer, 2012, pp. 35–85, ISBN: 978-1-4419-9326-7. DOI: 10.1007/978-1-4419-9326-7_2.

[49] J. H. Friedman, 'Greedy function approximation: A gradient boosting machine.,' *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001, Publisher: Institute of Mathematical Statistics, ISSN: 0090-5364, 2168-8966. DOI: 10.1214/aos/1013203451.

[50] W3Schools. 'Python round() function.' (2026), [Online]. Available: https://www.w3schools.com/python/ref_func_round.asp (visited on 14/06/2026).

[51] M. G. Lawrence, 'The relationship between relative humidity and the dewpoint temperature in moist air: A simple conversion and applications,' 1st Feb. 2005, Section: Bulletin of the American Meteorological Society. DOI: 10.1175/BAMS-86-2-225.

[52] X. Lin and K. G. Hubbard, 'Uncertainties of derived dewpoint temperature and relative humidity,' 1st May 2004, Section: Journal of Applied Meteorology and Climatology. DOI: 10.1175/2100.1.

[53] B. N. Taylor and C. E. Kuyatt, *Guidelines for evaluating and expressing the uncertainty of NIST measurement results*, NIST Technical Note 1297; Accessed: December 3, 2025, National Institute of Standards and Technology (NIST), 1994. [Online]. Available: https://physics.nist.gov/cuu/Uncertainty/index.html (visited on 03/12/2025).

Appendix A

The full data mining project is provided as online appendix for reproducability purposes. Due to copyright reasons, all input data files and training data derived from input data have been redacted to retain only their headers. Trained models, as well as target class files, have not been removed from the published repository. The complete project, including the redacted data files, can be requested from the publishing institute, or gathered manually by following the description of the data gathering process within the thesis.

- **Published Appendix:** https://doi.org/10.5281/zenodo.19099829