UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Study of delay prediction in the US airport network

MASTER FINAL THESIS

Document:

Report

Autor:

Kerim Kiliç

Director:

Dr. Jose Maria Sallan

Degree:

Master in technology and engineering management

Examination session

Autumn, 2022 - 2023

“Hayatta en hakiki mürşit ilimdir.”

“The truest guide in life is science.”

“La verdadera guía en la vida es la ciencia.”

- Mustafa Kemal Atatürk

Abstract

In modern business, Artificial Intelligence (AI) and Machine Learning (ML) have affected strategy and decision-making positively in the form of predictive modeling. This study aims to use ML and AI to predict arrival flight delays in the United States airport network. Flight delays carry severe social, environmental, and economic impacts with them, and deploying ML models during the process of strategic decision-making, can help to reduce the impacts of these delays.

To achieve the result of the study, a literature study and critical appraisal have been carried out on previous studies and research relating to flight delay prediction. In the literature study, the datasets used, selected features, selected algorithms, and evaluation tools used in previous studies have been analyzed. The results from the literature study and critical appraisal have influenced the decisions made in the methodology for this study. In the methodology, a choice is made for two public datasets, one of the domestic flight data of 2017 and one of the weather data of 2017. These two datasets are then processed in a custom-designed data pipeline which is built using Spark. The processed data is split into training data, validation data, and testing data. The training data and validation data are used to train and hyperparameter tune several ML models using both Spark and H2O. Subsequently, these ML models are evaluated and compared based on performance metrics obtained using the testing data. From this comparison, the best-performing model is presented as a suitable solution for arrival flight delay prediction.

The predictive model with the best performance among logistic regression, random forest, gradient boosting machine, and feed-forward neural networks ended up being the gradient boosting machine with far better predictive modeling performance. This solution can be deployed as a supportive ML model during strategic decision-making.

Resumen

En los negocios modernos, la inteligencia artificial (IA) y el aprendizaje automático han impactado positivamente la estrategia y la toma de decisiones a través del uso de modelos predictivos. Este estudio tiene como objetivo utilizar el aprendizaje automático *(machine learning)* y la IA para predecir los retrasos en las llegadas de los vuelos en la red de aeropuertos de los Estados Unidos. Los retrasos en los vuelos conllevan graves impactos sociales, ambientales y económicos, y la implementación de modelos de aprendizaje automático en el proceso de toma de decisiones estratégicas puede ayudar a reducir los impactos de estos retrasos.

Para conseguir el objetivo del estudio, en primer lugar se ha llevado a cabo un estudio bibliográfico y una valoración crítica de estudios e investigaciones previas relacionadas con la predicción de retrasos de vuelos. En la revisión de la literatura, se analizaron los conjuntos de datos utilizados, las características seleccionadas, los algoritmos seleccionados y las herramientas de evaluación utilizadas en estudios anteriores. La valoración crítica de los resultados de la revisión de la literatura han influido en las decisiones tomadas en la metodología de este estudio. Para el análisis, se han utilizado dos conjuntos de datos públicos, por un lado un listado de los de vuelos nacionales de 2017 y por otro los datos meteorológicos de 2017. Estos dos conjuntos de datos se han procesado mediante Spark mediante un *pipeline* de instrucciones en lenguaje R. Los datos resultantes se han dividido en datos de entrenamiento, datos de validación y datos de prueba. Los datos de entrenamiento y los datos de validación se usan para entrenar y ajustar hiperparámetros en varios modelos de aprendizaje automático usando Spark y H2O. Posteriormente, estos modelos de aprendizaje automático se evalúan y comparan en función de los valores para las métricas de rendimiento obtenidos a partir de los datos de prueba. A partir de esta comparación, el modelo de mejor rendimiento se presenta como una solución adecuada para la predicción de retrasos en los vuelos de llegada.

El modelo predictivo con el mejor rendimiento entre la regresión logística, el *random forest*, la *gradient boosting machine* y las redes neuronales de alimentación hacia adelante terminó siendo la la *gradient boosting machine* con un rendimiento de modelado predictivo mucho mejor. El modelo obtenido se puede implementar como un modelo de aprendizaje automático de apoyo para la toma de decisiones estratégicas.

Table of Contents

List of tables

List of figures

List of abbreviations

ML	Machine Learning
AI	Artificial Intelligence
US	United States
BTS	Bureau of Transport Statistics
NOAA	National Oceanic and Atmospheric Administration
NN	Neural Network
ISD	Integrated Surface Database
FAA	Federal Aviation Administration
GPS	Global Positioning System
ROC AUC	Area Under the Receiver Operating Characteristic Curve
PR AUC	Area Under the Precision-Recall Curve

Chapter 1

Introduction

1.1 Objective

The objective of this master thesis project is to study arrival flight delay prediction in the United States (US) airport network and propose multiple machine learning (ML) and artificial intelligence (AI) solutions for performing these predictions. The objective is to use data known before departure to predict the arrival delay. Data obtained from various open and public databases of the US are utilized to train a variety of ML models. These ML models are then compared based on performance and the ability to predict if a given flight will experience a delay yes or no. The objective is to propose a final model and solution from the various obtained ML models as the best for predicting flight delays.

1.2 Scope

The objective of the master thesis is to propose an ML model that is capable of predicting arrival flight delays in the US airport network.

In order to achieve the desired results, several data sources will be chosen to use in the delay prediction. The scope of this study is set to use the data from 2017. A data pipeline will be built for these sources of data, which will be used to process this data for usage in training the ML models. These ML models will be tuned for optimal performance. The scope of the work is set to use the data of domestic flights in the US airport network. After building this data pipeline several algorithms will be chosen and trained using the processed data. Each of these algorithms will then be compared based on performance and the best-performing algorithm will be picked as the proposed solution to predict arrival flight delays in the US airport network.

1.3 Requirements

Regarding the requirements for this project, the first requirement is set to only use the domestic flight data of the US, since the study is about predicting flight delays in the US airport network. The data used to train these ML models will be originating from databases that are open to the public. It is a requirement to only use data that is known prior to the departure of the specific flight. The main programming language to perform the study and the analysis is the R programming language. Besides the R programming language, two big data tools, Spark and H2O will be used to efficiently process large quantities of data. The R programming language will be used to interact with these big data tools. The code produced for this master thesis project will be kept public and available on a GitHub repository [1].

1.4 Rationale

Flight delays bring a significant negative impact on people and the economy. Besides the fact that experiencing a flight delay can be extremely troublesome for individuals during their travels, it has an even more significant economic impact. A study from 2013 [2] shows that a decrease of 10 percent in flight delays would increase the US net welfare by $17.6 billion and a decrease of 30 percent would increase the US net welfare by a staggering $38.5 billion. Besides the economic impact, flight delays have also a very negative impact on the environment. A study performed in 2018 [3] shows that in the year 2017, the emission due to flight delays was estimated at 5520 tonnes while excess fuel usage was estimated at 1.752.937 liters. The social, economic, and environmental impact makes it worth exploring the usage of AI and ML in order to predict flight delays and use these models to prevent these flight delays in the future.

Chapter 2

Literature review

This chapter consists of a review of previous works and studies performed around the topic of flight delay prediction. Different works will be critically analyzed for information that relates back to the present study. Information that will be extracted will be the data sources used, the features that are selected, the algorithms used to perform the predictions, and how the state of the art evaluates the performance of these models.

2.1 Datasets selection

The first set of information to extract from the state of the art is the datasets that are being used in previous research. After the datasets are each discussed, a critical appraisal will follow.

2.1.1 Datasets used

In Miguel Lambelho et al. [4] an attempt is done at predicting flight delays and cancellations at London Heathrow airport. Part of the data used in this study comes directly from the airport of London Heathrow, as well as from the European Organization for the Safety of Air Navigation [5], better known as Eurocontrol. Neither of these two databases is open to the public or provides open access.

In Sun Choi et al. [6] a ML model is proposed to attempt to predict arrival delays in flights. In this research, they obtained flight and weather data from 2005 to 2015. The flight data originates from the Bureau of Transport Statistics (BTS) [7] of the US Department of Transportation. The weather data originates from the National Oceanic and Atmospheric Administration (NOAA) [8] of the US Department of Commerce. Both of these sources of data are open and have public access making them relevant to the current research study. The data from 2005 to 2015 are used as training data to train the model and subsequently data from 2016 is used to evaluate the performance of this model as testing data. The target variable is chosen as arrival delay, which is simply calculated by subtracting the scheduled arrival time from the actual arrival time. Based on this delay time, the target variable is determined based on a threshold of 15 minutes. If the delay is equal to or greater than 15 minutes the flight is considered delayed, and if the delay is lesser than 15 minutes the flight is considered to be on time. Another detail to note here about this research study

is that there is only a single flight route chosen and the data is filtered by this route. In this study, only the flights departing from Denver International Airport flying to Charlotte Douglass International airport are considered. This narrows down the applicability in the wider US airport network. In Sun Choi et al. the imbalance between the number of delayed flights and the number of non-delayed flights is also pointed out, the dataset is highly imbalanced. Both undersampling and oversampling are used as a technique to overcome this balance issue and in the research, it is shown that undersampling improved the performance while oversampling made the performance worse.

In Peng Hu et al. [9] a ML model is proposed to predict arrival delays in the Chinese airport network. This research is carried out in collaboration with the Civil Aviation Administration of China [10], it is not stated where they obtained the data from, but this could very well be from the Chinese Civil Aviation Administration [10]. The data consists only of June 2020 and it is limited to flights departing from Guangzhou Baiyun International airport. The arrival airports are limited to the ten most significant landing airports that are connected to the departing airport. The final dataset that is being used consists of only 4947 flights and the number of delayed flights accounts for roughly 85% of the dataset used.

In Bin Yu et al. [11] a ML approach is proposed to predict numerical flight delays in minutes, in the Chinese airport network. The data originates from a closed database owned by Beijing capital airport. In this study, only flights between two major airports in China are considered, Beijing capital airport and Hangzhou international airport. The data contains flights between January 2017 and March 2018.

In Young Jin Kim et al. [12] artificial neural networks (NN) are proposed as a solution for flight delay prediction. In this study, weather data and flight data are combined together to predict arrival delays of flights in the US airport network. The flight data originates from the reporting carrier on-time performance database from the BTS of the US Department of Transportation [7]. The weather data originates from the integrated surface database (ISD) [8] of the NOAA which is part of the US department of commerce. The data that they are using spans almost 5 years from January 2010 up to august 2015. In the study the flight data is filtered to use only the data of ten major airports in the US. Both the flight and weather databases are public and provide open access.

In Wei Shao et al. [13] a ML approach is proposed for the prediction of departure delays. In this study, various sources of data are utilized, including flight data, weather data, and data on air traffic complexity. For the flight data, the study used the reporting carrier on-time performance database from the BTS of the US department of transportation [7]. The data covered only two months from the first of July 2016 to the 31st of August 2016. There is only one target airport which is flights originating from Los Angeles international airport. The departure delay is used as a target variable, and the prediction is done to the number of minutes of delay a given flight experiences prior to departure. The weather data originates from Weather Underground [14], which contains historical weather data of big cities in the US. The dataset on air traffic complexity originates from the US Federal Aviation Administration's (FAA's) System Wide Information Management program [15]. This is a private dataset with limited access. This dataset contains Global Positioning System (GPS) coordinates for different flight trajectories for different aircraft.

A summary of all of the data sources used in the state of the art is given in table 2.1. The summary contains the origin of the data, the type, and if the data is public for each corresponding research study.

Table 2.1: Summary of datasets used in state of the art

Research study	Data origin	Type	Public data?
Miguel Lambelho et al.	London Heathrow airport	Flight data	No
	Eurocontrol [5]	Airport data	No
Sun Choi et al.	BTS [7]	US flight data	Yes
	NOAA [8]	Weather data	Yes
Peng Hu et al.	Guangzhou Baiyun airport	Flight data	No
Bin Yu et al.	Beijing capital airport	Flight data	No
Young Jin Kim et al.	BTS [7]	US flight data	Yes
	NOAA [8]	Weather data	Yes
Wei Shao et al.	BTS [7]	US flight data	Yes
	Weather underground [14]	Weather data	Yes
	US FAA [15]	Trajectory data	No

2.1.2 Critical appraisal

In this section, a critical appraisal of the chosen datasets in the state of the art will follow. Since the current study focuses on delay prediction in the US airport network, it makes sense to apply this critical appraisal to the past studies that researched the possibility of deploying ML models to predict delays in the US. For the studies predicting flight delays in other geographical locations, a critical appraisal will be applied to the feature selection, and the algorithm and evaluation, which follows in the next sections.

Both Sun Choi et al. [6], Young Kim et al. [12], and Wei Shao et al. [13] obtain their flight data from the BTS [7]. This makes sense as this is data coming from a US government agency that is linked with the US Department of Transportation, making it a reliable source of flight data. The data is publicly available and has open access and can easily be downloaded from their website. In the current study, a similar approach will be taken for the dataset selection for the flights as the BTS provides reliable and complete data. Regarding the weather data used, both Sun Choi et al. [6] and Young Kim et al. [12] are using the data provided by the NOAA [8], whereas Wei Shao et al. [13] uses weather data from Weather underground [14]. Both these data sources seem reliable, the NOAA being part of the US Department of Commerce, whereas Weather Underground is a commercial source part of the industry. To keep sources of data consistent, the current study is chosen to pick sources of data that originate from the US government, therefore both the BTS, as well as the NOAA will be used to collect flight and weather data respectively. Besides weather and flight data, in Wei Shao et al. a private database from the FAA [15] is used for GPS trajectory data. Due to the inaccessibility of this database, since it is not open to the public, it will not be used in the current study.

Another interesting observation is that in all previous studies, except for Peng Hu et al. [9] and Young Jin Kim et al. [12], only a single flight route is chosen to apply predictions. In Peng Hu et al. [9] a single

departure airport is chosen and the flights are filtered based on the ten most significant landing airports. In Young Jin Kim et al. [12], both the departure airport and the arrival airport are filtered to be one of the ten most significant airports with the most flights. This approach of filtering the selected data allows for wider applicability of a model, instead of it only being applicable to a single route. In the current study, a similar approach will be taken to Young Jin Kim et al. [12], filtering the origin and destination airport with the most flights, keeping the ten most significant airports. Although the approach in Peng Hu et al. [9] is proper for applying to multiple flight routes, it would require multiple ML models to make a prediction for multiple departure airports, this is why an approach similar to Young Jin Kim et al.[12] will be followed in the current study, to apply a single ML model to multiple flight routes and airports.

And the last detail to point out is that in none of the previous studies, except for Sun Choi et al. [6] the imbalance of the classes of the target variable was discussed. Sun Choi et al. show that applying undersampling to the training data improves performance and that oversampling does not work in the case of the delay prediction problem. It is always a good practice to balance the training dataset in imbalanced data, and therefore in the current study undersampling in the training dataset will be applied as well.

2.2 Feature selection

The next set of information to extract from the state of the art is the features that have been selected to predict flight delays in previous research that has been carried out. This is followed by a critical appraisal of said features.

2.2.1 Features selected

In Miguel Lambelho et al. [4] the following features are used in a flight delay prediction algorithm for flights departing from London Heathrow airport. The airline, the terminal, the type of aircraft, the origin airport where it came from or destination airport where it is going to, the year in which the flight is scheduled, the month in which the flight is scheduled, the hour in which the flight is scheduled, the day of the year when the flight is scheduled, the day of the month when the flight is scheduled, the day of the week when the flight is scheduled, the distance between origin and destination airport, the country the flight came from or is going to, the number of seats in the aircraft, and the daily average of delay in the specific airport terminal.

In Sun Choi et al. [6] there are two types of features used, weather features and flight features. The weather features come from data from NOAA [8] and the flight features from BTS [7]. The flight features consist of the quarter in which the flight is scheduled, the month in which the flight is scheduled, the day of the month in which the flight is scheduled, the day of the week in which the flight is scheduled, the scheduled departure time and the scheduled arrival time. The weather features consist of the wind direction, the wind speed, the visibility, the precipitation, the weather intensity code, the weather descriptor, the obscuration code, the other weather code, and the combination indicator code.

In Peng Hu et al. [9] several features are described in the research that is being used to train the model and predict arrival delays. The first feature mentioned is the departure delay time. This is an unusual feature

to use to predict the target variable arrival delay since they are directly related to each other. The feature could still be used, however, it is not possible to predict any future flight delays before the aircraft takes off. Prediction using this feature is only possible after the aircraft takes off since departure delay is only known after the aircraft actually departs. The second feature mentioned is the planned flight time of a flight. The third feature is the total number of scheduled departure flights on the day of the flight, this is to model any congestion in the airport. The fourth feature is defined as the flight date, the attributes that include this are the scheduled day and month, and the final feature which is taken into consideration is the destination airport.

In Guan Gui et al. [16] multiple ML models are proposed to predict arrival delays in major cities in China. The research does not disclose the source of data, however, it discloses the features used. The first set of features is weather features. This includes the weather condition of the departure airport, the weather condition of the destination airport, the wind direction of the departure airport, the wind power of the departure airport, the wind direction of the destination airport, and the wind power of the destination airport, respectively. The second set of features is time and date-related features. In this research, they are using the day of the month, the month, the day of the week, and the season, respectively. And the last set of features are features related to the scheduling of the flight. This includes the departure airport, the scheduled time of departure, the destination airport, and the scheduled time of arrival, respectively.

In Bin Yu et al. [11] several flight and flight scheduling-related features are proposed to perform numerical prediction of flight delays between two major Chinese airports. One of the features is if the aircraft had a delay in its previous flight. This is determined by looking at the tailnumber of the flight and determining if this specific tailnumber had a delay in its previous flight. Another feature considered in this study is aircraft capacity or the number of seats in the aircraft. And the final feature used in this study is airport congestion, which is modeled by the total number of departing and arriving flights at a given airport on the scheduled flight date.

In Young Jin Kim et al. [12] several features are proposed to initially predict a day-to-day delay status flag, which is subsequently used to predict delays. It is unclear if the target variable is the arrival delay or if the target variable is the departure delay. The features proposed to predict the day-to-day delay status flag are divided into weather features and flight features. The weather features are the wind direction, wind speed, cloud heights, visibility, precipitation, snow accumulation, intensity, descriptor, and weather observation code. All of the weather-related features are averaged for each day. The flight data features are the top 10 origin and destination airports with the most flights, the day of the week in which the flight is scheduled, the season in which the flight is scheduled, the month in which the flight is scheduled, and the scheduled flight date itself. In order to predict the delay of individual flights similar features are used. The day-to-day flag which is predicted using the previous stage is one of the features. The remaining features are very similar to the features used to predict the day-to-day flag and can similarly be categorized into flight data features and weather data features. The flight features include the day of the week in which the flight is scheduled, the season in which the flight is scheduled, the scheduled month, the scheduled flight date, the origin airport, the destination airport, the scheduled departure time, and the scheduled arrival time. From the weather data,

the following features are included, the wind direction, the wind speed, the cloud height, the visibility, the precipitation, the snow accumulation, the intensity, the weather descriptor, and the weather observation code.

In Wei Shao et al. [13] several features are proposed that are used to train ML models to predict departure delays at the Los Angeles International airport. The features originate from three different sources, flight data, weather data, and from GPS data. The features from the flight data are the scheduled departure time, scheduled flight time, and scheduled arrival time. The weather features that are used in this study are wind direction, wind speed, ambient air pressure, wind gust, humidity, dew point, temperature, and the weather condition as a categorical feature. The third feature set is the air traffic complexity. air traffic complexity is calculated using GPS coordinates of different flight trajectories for different aircraft and models the congestion or complexity of air traffic or trajectories.

2.2.2 Critical appraisal

In this section, a critical appraisal of the chosen features in the state of the art will follow. Date and date-time-related features, such as the month, day of the month, and day of the week, seem to be chosen in almost all studies that are analyzed in the previous section. Except for Peng Hu et al. [9], Bin Yu et al. [11], and Wei Shao et al. [13] where these date and date-time features are missing. It could be argued that these features are important, as there could be more flight delays in specific months or on a specific day of the week for instance. In the current study, the date and date-time features will be explored for the prediction of flight delays. Another interesting observation from the literature research is that except for Peng Hu et al. [9] and Bin Yu et al. [11], airport congestion is not used in any of the other studies. Airport congestion should be a good feature to use, since the amount of departing and arriving flights at the origin or destination airport, could directly influence the number of delays. Another interesting observation from the literature study is that none of the studies takes the carrier into account, which arguably could make an impact when using it as a feature in flight delay prediction. There could be specific carriers that experience delays more often compared to other carriers, therefore this is a feature that will be taken into account in the current study. When looking critically at Peng Hu et al. [9], one thing that clearly came up in the literature study is the fact that the departure delay is used as a feature to predict the arrival delay. It would seem that using this as a feature is not proper, since the objective is to identify any flights that will experience a delay before takeoff, and the departure delay is not known until after takeoff. In Miguel Lambelho et al. [4] and Bin Yu et al. [11] the seating capacity of the flight, as well as the flight distance, are used, these two features are not used in any of the other studies. The seating capacity and flight distance are considered good features since with a larger flight distance and seating capacity, there could occur more passenger congestion in the airport, during boarding perhaps. Therefore this feature will be taken into account in the current study as well. When looking at the usage of weather features, Miguel Lambelho et al. [4], Peng Hu et al. [9], and Bin Yu et al. [11] are not using a single weather feature in their prediction strategy. One could postulate that the weather impacts flights and therefore the delay of flights as well. Therefore in the current study, all of the important features originating from weather data will be considered as features.

Two features that are not used in any of the previous studies are scheduled flight time and scheduled flight

speed. The scheduled flight time and scheduled flight speed could indirectly affect if a flight might experience a delay, as flights that have a greater flight time, might experience more delay upon arrival. Similarly, a flight that requires a greater speed to arrive in time might experience more delay at arrival. These two features will be considered in the current study as well in order to predict arrival delays.

2.3 Algorithm and evaluation

The next set of information to extract from the state of the art is the algorithm or algorithms that have been selected to train the prediction models, as well as how these models are evaluated. This is followed by a critical appraisal of the used algorithms and their evaluation methods.

2.3.1 Algorithms and evaluation methods used

In Miguel Lambelho et al. [4] three different model algorithms are considered and compared for flight delay prediction. LightGBM, which is a variant of gradient boosting machine, multi-layer perceptron which is a type of feed-forward NN, as well as random forest which is an ensemble of multiple decision tree algorithms.

Table 2.2: LightGBM model parameters in M. Lambalho et al. [4]

Classifier	**Learning rate**	**Max depth**	**Trees**
Departure delay	0.1	15	300
Arrival delay	0.1	39	200

The parameters set for the LightGBM model are the learning rate, max depth of each tree, and the number of trees and are summarised in table 2.2. For the multi-layer perceptron model, it is mentioned that an Adam optimized is used and that the parameters that are set are the number of neurons for each layer, the dropout rate, and the learning rate. These parameters are summarised in table 2.3.

Table 2.3: Multi-layer perceptron model parameters in M. Lambalho et al. [4]

Classifier	**Neurons for each layer**	**Dropout rate**	**Learning rate**
Departure delay	$100 \rightarrow 100$	0.15	$1.0 \cdot 10^{-4}$
Arrival delay	$110 \rightarrow 110$	0.05	$1.0 \cdot 10^{-3}$

For the random forest model, it is mentioned that the parameters that are set are the number of trees that are generated, the max depth of each tree, and the percentage of features for each split. These parameters are summarised in table 2.4.

Table 2.4: Random forest model parameters in M. Lambalho et al. [4]

Classifier	**Number of trees**	**Max depth of tree**	**% features for each split**
Departure delay	500	11	0.60
Arrival delay	1000	12	0.55

Each model was cross-validated using 5-fold. To compare these three models in the study following evaluation metrics were used to evaluate and compare the models. The accuracy, precision, recall, the F1 score, and the area under the receiver operating curve (ROC AUC). Besides the metrics to evaluate the models, the computational time is also evaluated for each model. Of the three mentioned models LightGBM performs the best with respect to the accuracy, precision, recall, and ROC AUC. The LightGBM model had an accuracy of 79%, precision of 57%, recall of 55%, F1-score of 56%, and ROC AUC of 80%. It is mentioned that the models are evaluated using test data.

In Sun Choi et al. [6] four different algorithms are proposed and compared based on model performance. Decision trees, which is a commonly used algorithm for classification tasks, random forest, which is an ensemble of multiple decision trees, adaptive boosting, which is a statistical model used most commonly for binary classification such as the flight delay problem, as well as a k-Nearest Neighbours model, which is another model algorithm that works well with classification jobs as well as regression jobs. Not a lot of details are mentioned regarding the settings or parameters of each model, for the tree-based models it is mentioned that the maximum depth is set at 6 and for the random forest, the number of trees is set at 100, for the AdaBoost model, the learning rate was selected at 1 and the maximum number of estimators at 100, and for the k-Nearest Neighbours model, the number of neighbors was selected at 6. Nothing is mentioned about any form of model tuning, however, all the models were cross-validated using 10-fold cross-validation. To compare the four models, several evaluation metrics are used based on the training data. The model accuracy, the true positive rate, the false positive rate, and the ROC AUC. Since the delay dataset is highly imbalanced, in this research more emphasis is put on the ROC AUC as the performance metric. The models were also evaluated on test data, but with the test data, only accuracy was used as the metric. In the final conclusion, the random forest had the best performance based on accuracy from all the models that were presented in this study. The random forest model had an accuracy of 80% on the test set.

In Peng Hu et al. [9] a random forest model is proposed to predict arrival delays. Regarding the settings of the algorithm, two parameters have been tested to find optimal values and tune the model, the leaf size, and the number of trees. A series of tests showed that the optimal leaf size in their case study was 5 with the optimum number of trees set at 100. No other parameters of the random forest model were tuned and assumed to be set at the default values. In order to evaluate the performance of the random forest model the study looks at accuracy which is obtained from training data. Nowhere in the study is a training and testing data split mentioned to perform the proper model evaluation. The accuracy obtained from the training data is very high, however, it is not possible to exclude overfitting, as there is no testing done on an isolated testing data set. The accuracy obtained from the training data is equal to 90%.

In Guan Gui et al. [16] two classification models are proposed to predict arrival delays in the Chinese airport network. Any arrival delay that is equal to or larger than 15 minutes is classified as a delay, and any delay smaller than 15 minutes is considered not delayed. The first model is a long short-term memory artificial NN. This model algorithm is part of a broader class of recurrent NNs that work very well for data with temporal features. From this model algorithm, three different architectures are chosen to predict arrival delays, the standard architecture, the architecture that utilizes a fully connected layer, and the architecture that utilizes

a dropout layer. This model has been tuned with a grid-search method with the number of epochs set to 1500, 2000, and 3000. And the memory depth is tuned with a grid search defined as 3, 5, and 7. The second model that is explored is the random forest model, which is a combination of multiple decision trees that work very well with classification problems. To the random forest model, a grid search has been applied to the number of trees, the maximum depth of each decision tree, and the maximum number of features per decision tree. The maximum number of trees was tuned with values between 20 trees and 150 trees. The maximum depth of the decision tree was tuned with values between 3 and 20. And the maximum features per decision tree were tuned with the second logarithm function of the total number of features in the dataset, and the square root of the total number of features in the dataset. In the final model, 35 trees were chosen, with a max depth of 12, and the maximum number of features per decision tree was determined by the square root of the total number of features in the dataset. These two models are then evaluated by looking at the model accuracy on the training data and comparing this with the testing data. One of the conclusions of the work is that random forest is a suitable method for a classification job in this case but using the long short-term memory artificial NN causes a lot of overfitting which makes it unsuitable for a classification job like this one. The long short-term memory NN obtains 88% accuracy on the training set but just 37% on the testing set. With the random forest model, it is stated that on the training set accuracy of 90% is achieved and it is mentioned that on the testing set 70% accuracy is obtained. However no details for this data split are mentioned in the study.

In Bin Yu et al. [11] an architecture is proposed that consists of two algorithms. The first algorithm is a deep belief network that uses unsupervised learning to find patterns to extract features from the data provided at its input. Subsequently, these features are then fed into the second stage which is a support vector regressor, which uses the extracted features to predict a numerical delay in the number of minutes for each flight. The proposed architecture is evaluated by using the mean absolute error in the number of minutes. Mean absolute error is a commonly used evaluation metric to benchmark how well a numerical model is performing. Even though the current study performed is a classification problem, the features and architecture used in this numerical delay study could come in useful during further research and application in classification-related studies.

In Young Jin Kim et al. [12] a two-staged ML solution is proposed to predict flight delays. The first stage consists of a Recurrent NN, which predicts a day-to-day delay status flag for different airports. The day-to-day delay status flag is simply defined as the average delay any given airport will experience across all of its flights, which is categorized as delays larger or equal to 15 minutes or delays smaller than 15 minutes. In the second stage, a regular feed-forward NN is used to predict the delay of the individual flights, using the output of the first stage as the input to the second stage. In the second stage, they are using a sigmoid function as the transformation function, since this study deals with binary classification, and uses up to 5 hidden layers in their neural network. In order to evaluate the performance of the models they are using accuracy as the performance metric. For the day-to-day delay status model in the first stage, the accuracy is calculated for each of the top 10 airports. For the second stage, the feed-forward NN accuracy is used to compare the performance of the model when using a different number of layers up to 5. The accuracy increases only with

2% when increasing the number of layers from 1 to 5, from 85% to 87%. Model evaluation using a testing set is not mentioned anywhere. The different constructions of layers, with the number of nodes and epochs, are summarized in table 2.5.

Table 2.5: Layers, nodes, and epochs in Y. J. Kim et al. [12]

Layers	**Nodes per layer**	**Number of epochs**
1	133	22
2	$133 \rightarrow 100$	22
3	$133 \rightarrow 200 \rightarrow 15$	22
4	$133 \rightarrow 200 \rightarrow 100 \rightarrow 15$	22
5	$133 \rightarrow 300 \rightarrow 200 \rightarrow 100 \rightarrow 15$	22
5	$133 \rightarrow 300 \rightarrow 200 \rightarrow 100 \rightarrow 15$	228

In Wei Shao et al. [13] three algorithms are considered and compared for the numerical prediction of departure delays. Light gradient boosting machine, which is a popular ML algorithm and effective in building predictive models. For the light gradient boosting machine model, the parameters that were set are the learning rate, the number of estimators, and the number of leaves. The parameters are summarised in table 2.6. The usage of a support vector regressor is considered, which is a model that is suitable for regression tasks. For the support vector regression model, the parameters that were set are the kernel type and the error term. The parameters are summarised in table 2.6. A multi-layer perceptron is considered, which is a feed-forward NN. For the multi-layer perceptron model, the parameters that were set are the number of epochs, the early stopping round, the number of layers, the number of nodes, and the Adam learning rate. The parameters are summarised in table 2.6. These four algorithms are used to train the models which are then evaluated based on root mean square error. Root mean square error is a typical error-based evaluation metric, the lower the value, the better the model performs. From the four aforementioned models, the light gradient boosting machine has the lowest root mean square error and therefore the best performance.

2.3.2 Critical appraisal

In this section, a critical appraisal of the chosen algorithms and the evaluation methods in the state of the art will follow.

Chosen algorithms

The first point to evaluate is the different algorithms used in order to perform flight delay prediction. One of the algorithms that are used in multiple previously performed studies is the feed-forward NN. This type of algorithm is used in Miguel Lambelho et al. [4], Young Kim et al. [12], and Wei Shao et al. [13]. Usually for tabular data NNs are outperformed by tree-based algorithms like a random forest or a gradient-boosting machine model. Tree-based algorithms tend to work better with tabular data, these algorithms are explored in Miguel Lambelho et al. [4] in the form of a gradient-boosting algorithm, in Peng Hu et al. [9] as a random forest classifier, a random forest classifier is used for flight delay prediction in Guan Gui et al. [16] as well,

Table 2.6: Model parameters in W. Shao et al. [13]

Model	Parameter	value
LGBM	Learning rate	0.01
	Number of estimators	16000
	Number of leaves	39
SVR	Kernel type	rbf
	Error term	10000
MLP	Number of epochs	3000
	Early stopping round	50
	Number of layers	2
	Number of nodes	1553
	Adam learning rate	0.00976563

and Wei Shao et al. [13] also explores the usage of gradient-boosting machines. In the comparison between gradient boosting and feed-forward NN in Wei Shao et al. [13], the gradient boosting machine outperforms the NN. As mentioned previously, tabular data is usually a better fit for tree-based algorithms rather than NN-based algorithms.

In Sun Choi et al. [6] both a decision tree and a random forest algorithm are used, besides the k-nearest neighbors and the adaptive boosting algorithm. In the current study, from these models, only a random forest will be used, since a random forest is a combination of multiple decision trees, and multiple decision trees have better performance than a single decision tree. Neither adaptive boosting nor k-nearest neighbors are available in Spark or H2O, the tools used in this study, therefore these algorithms are not considered. In the study it is also shown that these two algorithms are outperformed by the random forest model.

None of the previous studies explores the usage of the logistic regression model for flight delay prediction, in order to make a proper comparison between different model types, the logistic regression model will be explored in this study as well. In order to have a proper comparison between well-performing models, logistic regression, random forest, gradient-boosting machine, and a feed-forward NN will be explored in the current study. This will allow for a good comparison of each model algorithm.

When looking at using cross-validation, only Sun Choi et al. [6] and Miguel Lambelho et al. [4] apply this technique with a number of folds of 5 and 10 respectively. Cross-validation is a standard tool that should be used when developing ML models and can be good to detect any overfitting or other problems in the model from occurring. Since the code in the current study is developed on a laptop with limited resources, the choice is made for a 4-fold cross-validation. This keeps the development time at an appropriate value.

Evaluation methods applied

Moving further to the model evaluation of the classification models, leaving out the studies that explored numerical prediction such as Wei Shao et al. [13] and Bin Yu et al. [11] since the evaluation methods in these

studies do not apply to classification. In some studies, it is very unclear if a train test data split has been used, therefore it is difficult to say if their model configuration contains any form of overfitting and if their model is properly trained. In most of the studies, no testing set is mentioned and only evaluation metrics based on the training data are given. This does not give a proper overview of any model performance at all. So in the current study, there will be a clear train-test split and even a train-validation-test split used, to be able to properly evaluate all of the models that will be trained. With regard to evaluation metrics, some studies only use accuracy as an evaluation metric, such as Peng Hu et al. [9], Guan Gui et al. [16], and Young Kim et al. [12]. It is important to use different model evaluation metrics to have a proper insight into the performance of the model. Therefore it is important to also use the ROC AUC, as used in Miguel Lambelho et al. [4] and Sun Choi et al. [6]. Other important evaluation metrics are precision, recall, and the F1 score, which are only used in Miguel Lambelho et al. [4]. In order to have a proper insight into model performance, in this study all the important model metrics will be evaluated when picking the best-performing model. These metrics will include, accuracy, recall, precision, F1 score, and ROC AUC. Besides these metrics also the area under the precision-recall curve (PR AUC) will be used, as this is an important metric to use in imbalanced datasets, as well as specificity, which gives an indication of the performance of detecting true negatives. Both of these metrics have not been used by any of the previous studies. Also, the visual representation of the PR and ROC curves will be compared for an even better comprehension of model performance.

Chapter 3

Methodology

This chapter consists of the methodology and steps used in order to reach the objective of the research study. First, an overview is given of the data that is used. This is followed by the entire ML process that needs to be followed to train the ML models as well as the steps to evaluate them. And the chapter is concluded with an overview of the two main tools used in this study Spark and H2O.

3.1 Data used

There are two sources of data considered in training the models and modeling the flight delay problem: the flight data itself and the weather data for each airport. In this section, the raw data and its attributes are discussed.

3.1.1 Flight data

The main source of data comes from the US Department of Transportation from a database of the BTS [7]. The flight data is obtained from the "Reporting Carrier On-Time Performance" database which contains detailed data on domestic flights from 1987-present. Within the scope of this study, the data used in the analysis comes from the entire year of 2017. The dataset contains 5665109 flights and 35 columns or attributes. The size of the data is 1.1 GB in comma-separated value format. The raw data consists of details of the date of each scheduled flight. Each variable or attribute related to the date and time can be found in table 3.1.

Table 3.1: Date and time attributes in BTS raw data [7]

Variable name	Details	Type	Example
year	The scheduled year of the flight	Integer	2017
quarter	The scheduled quarter of the flight	Integer	1, 2, 3, 4
month	The scheduled month of the flight	Integer	1, 2, 3, .. 12
day_of_month	The scheduled day of the month	Integer	1, 2, 3, .. 31
day_of_week	The scheduled day of the week	Integer	1, 2, 3, .. 7
hour_of_day	The scheduled hour of the day	Integer	1, 2, 3, .. 23
minutes_of_hour	The scheduled minute(s) of the hour	Integer	0, 1, 2, 3, .. 59

There are also variables and attributes present in the dataset related to the actual and scheduled arrival and departure time. These attributes are summarised in table 3.2 with the description, the type, and an example.

Table 3.2: Flight schedule attributes in BTS raw data [7]

Variable name	Details	Type	Example
planned_departure_time	Planned departure date and time.	Date-time	2017-01-01 01:00:00
planned_departure_local_hour	Planned departure local hour.	Integer	0, 1, 2, 3, .. 23
planned_arrival_time	Planned arrival date and time.	Date-time	2017-01-02 02:00:00
planned_arrival_local_hour	Planned arrival local hour.	Integer	0, 1, 2, 3, .. 23
actual_departure_time	Actual departure date and time	Date-time	2017-01-03 03:00:00
actual_departure_local_hour	Actual departure local hour.	Integer	0, 1, 2, 3, .. 23
actual_arrival_time	Actual arrival date and time	Date-time	2017-01-04 04:00:00
actual_arrival_local_hour	Actual arrival local hour.	Integer	0, 1, 2, 3, .. 23
wheels_off_time	Actual date and time of flight take off.	Date-time	2017-01-05 05:00:00
wheels_off_time_local_hour	Actual hour of flight take off.	Integer	0, 1, 2, 3, .. 23
wheels_on_time	Actual date and time of flight landing.	Date-time	2017-01-06 06:00:00
wheels_on_time_local_hour	Actual hour of flight landing.	Integer	0, 1, 2, 3, .. 23

There are also variables and attributes present for each specific flight and its aircraft. These variables are summarised in table 3.3, with the description, variable type, and an example.

Table 3.3: Flight detail attributes in BTS raw data [7]

Variable name	Details	Type	Example
carrier	Identifier of the flight carrier.	String	AA, AS, B6, DL, EV
tail_number	Identifier of the airplane.	String	N001AA, N104AA, N10575
flight_number	Identifier of the flight.	Integer	2330, 1590, 1320, 2202
origin	The origin airport.	String	ABE, STX, LAX, ORD
destination	The destination airport.	String	STS, SMF, SUN, RSW
cancelled_flag	Flag indicating if the flight was canceled.	Integer	0, 1
diverted_flag	Flag indicating if the flight was diverted.	Integer	0, 1
num_flights	The number of flights.	Integer	1
flight_distance	The distance of the flight in km(s).	Integer	650, 482, 518, 510
seating_capacity	Seating capacity of the airplane.	Integer	140, 196, 186, 176

And lastly, the dataset of flight information from 2017 also includes some potential target variables for different types of delays. These potential target variables include delays caused by the carrier, delays caused by the weather, delays caused by the national airspace system, delays caused by security, and delays caused by a late aircraft. These delays are summarised in table 3.4 below.

Table 3.4: Different delay categories in BTS raw data [7]

Variable name	Details	Type	Example
carrier_delay	Delay time caused by the carrier in minute(s).	Integer	0, 4, 13, 23
weather_delay	Delay time caused by the weather in minute(s).	Integer	2, 14, 20, 26
nas_delay	Delay time caused by national airspace in minute(s).	Integer	15, 16, 13, 14
security_delay	Delay time caused by security in minute(s).	Integer	2, 3, 4, 5
late_aircraft_delay	Delay time caused by late aircraft in minute(s).	Integer	7, 8, 9, 10

3.1.2 Weather data

The main source of the weather data comes from the ISD of the National Centers for Environmental Information of the NOAA [8]. The database for the weather data contains the average weather data for each day for different locations, including airports in the US. The variables and weather attributes are described in table 3.5 including their type and an example. The weather data used in this research study consists of the entire year of 2017 to complement the flight data which also consists of the entire year of 2017.

Table 3.5: Weather data in NOAA dataset [8]

Variable name	Details	Type	Example
Geographic location	Location of the weather station.	String	ORD, ATL, LAS, LAX
Date time	Date and time of the measurement.	Date-time	2017-02-02 14:00:00
Wind speed	The speed of the wind.	Double	3.58, 4.56, 6.71
Wind direction	The direction of the wind.	Double	118.31, 91.77, 209.13
Air temperature	The temperature of the air.	Double	8.23, 11.25, 15.33, 10.87
Atmospheric pressure	The atmospheric pressure at the location.	Double	1019.21, 1020.42, 1011.99
Visibility	The visibility at the given location.	Double	5042.91, 871.25, 10840.76
Dew point	The dew point at the given location.	Double	7.54, 11.15, 13.73, 7.20
Precipitation	The level of precipitation at the given location.	Double	2.82, 3.36, 4.81, 2.89
Cloud cover	The intensity of the clouds at the given location.	Integer	100, 85, 66, 74, 12
Wind gust	Increase in the wind speed at the given location.	Integer	3, 12, 24, 30
Total snow	Total snowfall at the given location.	Double	0.0, 22.6, 7.4, 5.4

3.2 Machine Learning process

In this section, the entire ML process is described from start to end. First, a high-level overview is given for the entire process. Followed by a more in-depth description of each part of the process.

3.2.1 High level overview

The ML process consists of several building blocks. The first step of the process is to take the raw data and transform and process it into useful data. This is followed by splitting the data into train and test datasets that are isolated from each other. The training dataset is in turn used to train each ML model and the test dataset is used on the trained models to evaluate the performance of the models. The input to this process is the raw data obtained from the databases described in the previous section and the output of the process is a ML model with its performance. This performance can then in turn be compared across different ML algorithms.

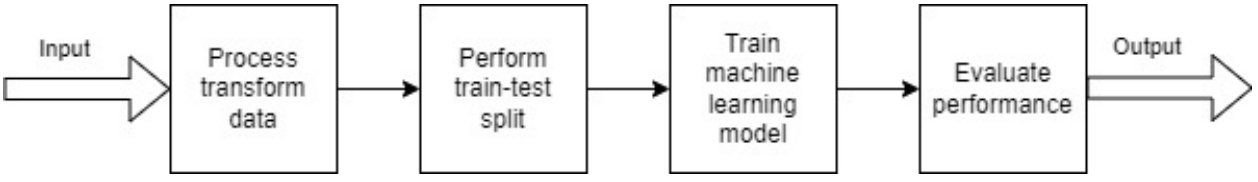

Figure 3.1: High level overview

3.2.2 Processing raw data

In order to make use of the raw data obtained from the databases previously mentioned it is necessary to process this data into a useful form. In order to process this data a data pipeline is designed. A high-level overview of the data processing steps is given in figure 3.2. The first step is to process the raw flight data mentioned in table 3.1, table 3.2, table 3.3, and table 3.4. The processed flight data is then merged with the weather data shown in table 2.1 and the airport congestion which is calculated later. This is followed by applying one-hot encoding to any of the categorical variables in the resulting merged dataset. And the last step is to remove any rows that contain missing (NA) values.

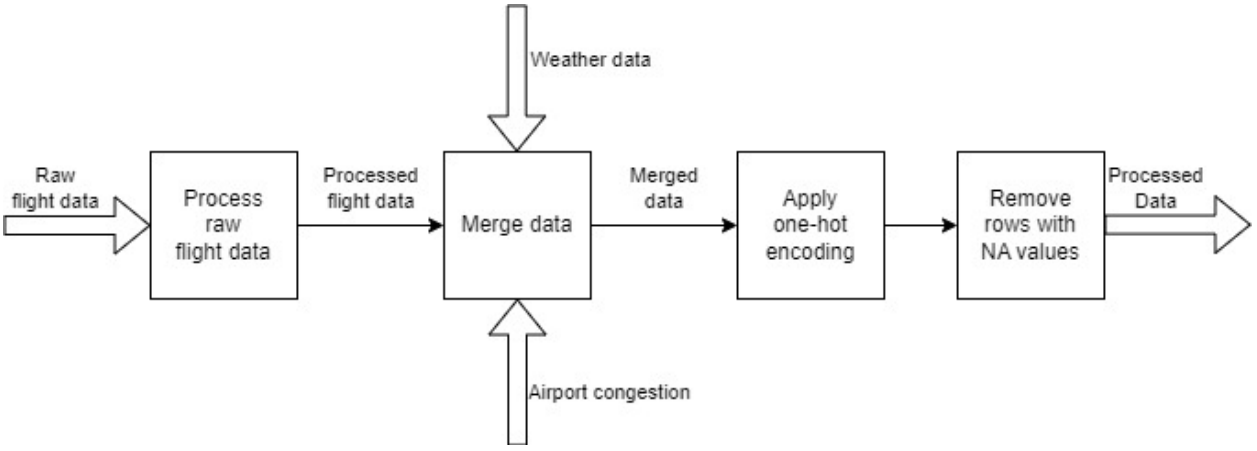

Figure 3.2: Data processing pipeline

The first step of processing the raw flight data in figure 3.2 is shown in more detail in figure 3.3.

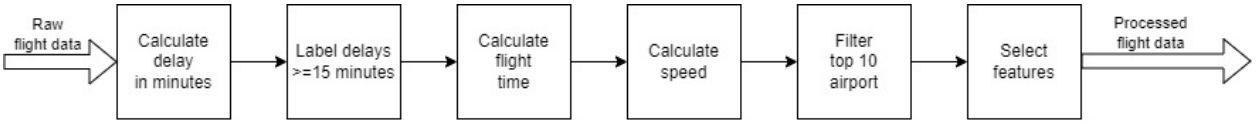

Figure 3.3: Data processing of raw flight data

The first two steps of processing the raw data are related to calculating and labeling the target variable. The target variable is the arrival delay, as such, there is a simple equation that can be applied to calculate this target variable. As definition per the BTS [17] any flight with a delay equal to or greater than 15 minutes is considered delayed, and any flight with a delay smaller than 15 minutes is considered on time. Therefore with the actual arrival time and planned arrival time from the raw flight data and using equation 3.1 the target variable can be calculated and labeled. A label "1" indicates a flight that is delayed and a label "0" indicates a flight that was on time.

$$\begin{aligned} (actual\ arrival\ time - planned\ arrival\ time) \geq 15\ minutes \longrightarrow delay = \text{"1"} \\ (actual\ arrival\ time - planned\ arrival\ time) < 15\ minutes \longrightarrow delay = \text{"0"} \end{aligned} \tag{3.1}$$

The next step is to add a feature which is the total flight time in minutes, which is calculated by simply subtracting the planned departure time from the planned arrival time. The calculation is shown in equation 3.2.

$$Flight\,time = (planned\,arrival\,time - planned\,departure\,time) \tag{3.2}$$

The following step is to calculate the flight speed, which is calculated by dividing the flight distance by the flight time which was calculated in the previous step. The calculation is shown in equation 3.3

$$flight\,speed = \frac{flight\,distance}{flight\,time} \tag{3.3}$$

Since the raw data might contain flights coming from or departing to very small airports that have a low number of flights, it is important to filter the data based on only the larger airports. This being said the next step is to filter out the top ten airports with the most flights. The airports with only very few flights are contaminating the dataset with noise. Therefore to filter out these small airports and only keep the ten airports with the most flights are taken into consideration in the dataset. Therefore it is possible to train a model which is applicable to the top ten airports with the most flights in the US airport network. The top ten airports and their symbol are given in table 3.6.

Table 3.6: Top ten airports with the most flights

Symbol	**Airport**	**State**
ATL	Hartsfield-Jackson Atlanta International Airport	Georgia
DEN	Denver International Airport	Colorado
DFW	Dallas/Fort Worth International Airport	Texas
LAS	Harry Reid International Airport	Nevada
LAX	Los Angeles International Airport	California
MSP	Minneapolis-Saint Paul International Airport	Minnesota
ORD	Chicago O'Hare International Airport	Illinois
PHX	Phoenix Sky Harbor International Airport	Arizona
SEA	Seattle-Tacoma International Airport	Washington
SFO	San Francisco International Airport	California

And the final step of processing the raw flight data is to select the variables and attributes from the processed data. This data will then be merged with additional data on the weather and airport congestion. The selected variables and features from the processed flight data are shown in table 3.7.

After selecting the features from the flight's dataset, the resulting dataset is joined together with the weather data for both the origin and destination airports in table 3.5 as well as with the airport congestion, this process is shown in figure 3.2. Airport congestion is a sub-dataset calculated from the raw data of the flight dataset. Airport congestion is a way to model how busy the origin and destination airports are at a given hour or day for the departure or arrival of a flight. In order to calculate airport congestion, all of the arriving

Table 3.7: Selected variables processed flight data

Variable name	Details	Type	Example
Delay	Target variable if a flight is delayed or not	String	"1" or "0"
Quarter	The scheduled quarter of the flight	Integer	1, 2, 3, 4
Month	The scheduled month of the flight	Integer	1, 2, 3, .. 12
Day of month	Day of the month in which the flight took place.	Integer	1, 2, 3, .. 31
Day of week	Day of the week in which the flight took place.	Integer	1, 2, 3, .. 7
Flight distance	Distance between the origin and destination.	Integer	650, 482, 518, 510
Seating capacity	The maximum seating capacity of the flight.	Integer	140, 196, 186, 176
Origin	Origin airport of the flight.	String	ATL, LAS, LAX
Destination	Destination airport of the flight.	String	ATL, LAS, LAX
Flight time	Total scheduled flight time of the flight.	Integer	60, 120, 135, 85
Flight speed	The scheduled average speed of the aircraft	Double	3.93, 5092.3
Carrier	Carrier number of the flight	String	AS, B6, DL, EV
Planned departure local hour	Planned departure hour of the flight.	Integer	0, 1, 2, 3, .. 23
Planned arrival local hour	Planned arrival hour of the flight.	Integer	0, 1, 2, 3, .. 23

and departing flights are summed up for each day for each specific airport. This is then added together to give a total number of flights per day per airport for both the origin and the destination airports. The equation for the calculation of airport congestion on a specific day for a specific airport X is shown in equation 3.4.

$$Airport\ congestion = (total\ arriving\ flights_{at\ airport\ X} + total\ departing\ flights_{at\ airport\ X}) \tag{3.4}$$

The penultimate step in the data processing steps, as shown in figure 3.2 is to apply one-hot encoding to all of the categorical variables. One-hot encoding is necessary since machine learning models only understand numerical values. One-hot encoding is a technique to transform a categorical value into a numerical value. This is done by creating a number of columns equal to the number of categories a categorical variable has. The value for this column can then be either 1 or 0, 1 if it belongs to this category, and 0 if it does not. There are only three categorical variables for which it is necessary to apply one-hot encoding: the origin airport, the destination airport, and the carrier. After performing one-hot encoding the final step is to remove any rows that contain any NA or empty value in any of the columns. NA or empty values in any of the columns of the dataset will cause errors when training or evaluating the ML models. This concludes the first step of the ML process shown in figure 3.1.

3.2.3 Train test split

Before proceeding to train the ML models it is first essential to split the data into so-called training and testing data sets. The training dataset is used to train each ML model, keeping the testing dataset isolated to evaluate the performance of the model. After training the ML model, the testing data is fed into the model, and predictions are made based on the testing data. The training and testing data are kept isolated in order to properly evaluate the performance of the model on unseen data. Usually, a majority of the processed data goes into the training data and a minority into the testing data. In the train-test split, a ratio of 90% for the

training data and 10% for the testing data is preserved. The process of train test splitting is shown in figure 3.4.

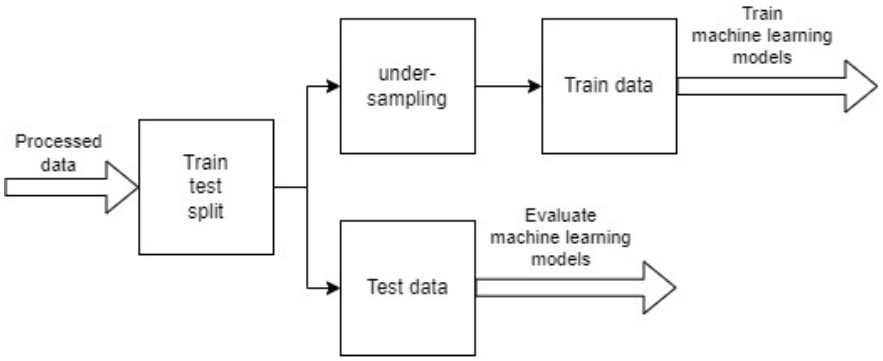

Figure 3.4: Train test split

The flight delay dataset can be imbalanced, meaning that there can be many more non-delayed flights than delayed flights. With imbalances such as these, predicting the minority class can be difficult, and performance can be improved by either undersampling or oversampling the majority class. Undersampling is chosen as the method to deal with class imbalances as Sun Choi et al. [6] showed that oversampling is not effective. In undersampling, there are simply rows randomly removed from the majority class until the training dataset is balanced. Undersampling is only applied to the training data as shown in figure 3.4.

In this study two distributed computing tools are used which are Spark and H2O, which will both be discussed in the tools section. H2O natively supports a validation set as well, which results in a train, validation, and test split. The validation set is used when the models are being cross-validated. This is useful because it isolates the data which is used to train the model, the data which is used to evaluate the models in cross-validation, and the data to evaluate the final model. The process of splitting data between a training set, validation set, and testing set is shown in figure 3.5. For H2O, the training data is undersampled as well. Both for H2O and Spark, the data composition in the testing set and in H2O for the validation set will be untouched, only modifying the training set, with the undersampling technique.

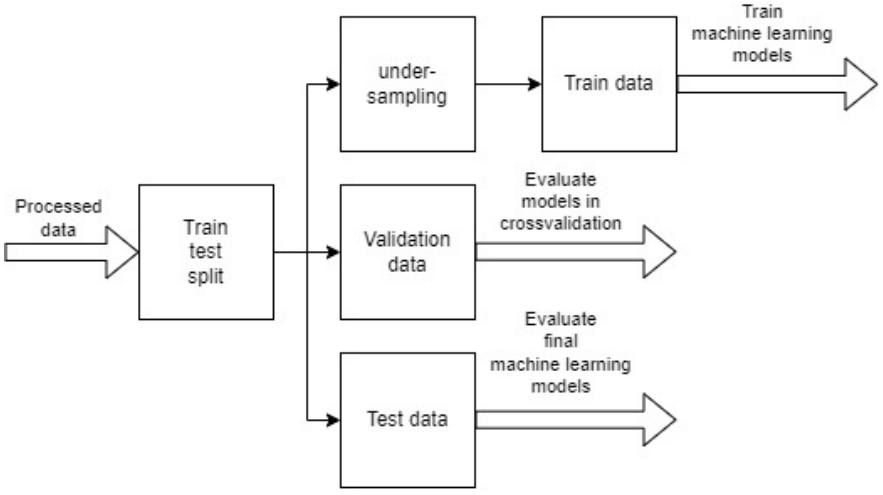

Figure 3.5: Train validation test split

3.2.4 Algorithms

Within the space of ML, there are several different algorithms to choose from. In order to have a good comprehension of model performance and to be able to make a good comparison, a variety of model algorithms is chosen to train and make predictions with. All these model algorithms and their performances will be compared in the results section that will follow later in this document. The four model algorithms chosen are logistic regression, random forest, both of which are trained using the Spark tool, and gradient boosting machine, and a feed-forward NN, both of which are trained using the H2O tool. Table 3.8 gives an overview of all the model algorithms that are considered and their corresponding tool. Each of these algorithms will be discussed in more detail and so will the tools used to train these algorithms.

Table 3.8: Algorithms considered in this study

Algorithm	**Tool**
Logistic regression	Spark
Random Forest	Spark
Gradient Boosting Machine	H2O
Feed Forward Neural Network	H2O

Logistic regression

Logistic regression, which is part of a broader class of models known as generalized linear models, is suitable for binary classification without very high hardware specifications [18]. Logistic regression has the benefit that it does not need very high hardware specifics in order to train models as mentioned before. In the literature, there are no studies that explored the usage of logistic regression for predicting flight delays which makes it interesting to test and explore. There are two parameters that can be tuned for logistic regression in Spark which are the elastic net regularization, also denoted as λ, and the regularization parameter also denoted as α [19]. Both the elastic net regularization as well as the regularization parameters have a minimum value of 0 and a maximum value of 1. Using hyperparameter tuning the optimal values for these parameters can be found and the model can have the optimal performance.

Random Forest

Random forest is another algorithm that makes use of an ensemble of multiple decision trees. The random forest algorithm has the benefit that it can capture non-linearities, prevent overfitting from taking place when carefully selecting variables, as well as it can work well with imbalanced data [20], which is the case for the delay prediction as explained in Sun Choi et al. [6]. Usage of the Random forest algorithm for flight delay prediction has been done by Sun Choi et al. [6], Peng Hu et al. [9], and Guan Gui et al. [16]. Two significant parameters which impact the performance are the depth of the tree and the number of trees [21]. Both the depth of the tree as well as the number of trees trained can be tuned using Spark [22]. Using hyperparameter tuning the optimal values for these parameters can be found and the model can have the optimal performance.

Gradient Boosting Machine

Gradient boosting is another commonly used ML algorithm for classification jobs. Gradient boosting in particular proves to work very well in big data and imbalanced datasets [23], which is the case for delay prediction as shown in Sun Choi et al. [6]. Gradient boosting is a method explored in Miguel Lambelho et al. [4] and in Wei Shao et al. [13]. The gradient boosting machine in H2O allows the tuning of the following parameters [24]: the maximum depth of each tree, the row sample rate, the column sampling rate per split, the column sampling rate per tree, the column sample rate change per level, the minimum rows, the number of bins for the numerical columns, the number of bins for the categorical columns, the minimum required error for a split, and the histogram type. These parameters will be tuned in order to obtain the most optimized model.

Feed Forward Neural Network

Feed-forward NNs are a type of Artificial NN that is becoming increasingly popular within the space of data science due to its ability to generalization [25]. This type of Artificial NN is also widely used in classification problems and can be used in tabular data as well [26]. Types of feed-forward Artificial NNs are explored an used in Miguel Lambelho et al. [4], Young Jin Kim et al. [12], and Wei Shao et al [13]. The following parameters can be tuned in H2O for feed-forward neural networks, the number of hidden layers, the number of nodes for each hidden layer, the input dropout ratio, the learning rate, and the learning rate annealing. These parameters will be tuned in order to obtain the most optimized model.

3.2.5 Cross-validation

Cross-validation is a method widely used in ML to evaluate the performance of classification algorithms [27]. The idea is to split the training dataset into K-folds with roughly an equal number of rows in each fold. The model is fit for K-1 and then predictions are made with fold K to obtain accuracy and to see the variability in the accuracy [28]. In order to reduce the time it takes to an appropriate value it is chosen to use 4-fold cross-validation. In the models trained using the Spark framework, there will not be a validation test set used as outlined in figure 3.4, since this is not supported natively in Spark [29]. H2O however does natively support the usage of validation sets in cross-validation [30], and therefore a validation set will be used as outlined in figure 3.5. Cross-validation is used in the research carried out in Miguel Lambelho et al.[4], where a 5-fold was used, as well as in Sun Choi et al. [6], where it was chosen to use a 10-fold cross-validation.

3.2.6 Evaluation using test data

It is important to split and isolate the data that is used to train ML models and the data that is subsequently used to evaluate these models [31]. The reason for this is that these models should be evaluated on unseen data and not on data that the models already have seen. The train test split and isolating these datasets is outlined in figure 3.4 and figure 3.5. The test data is used to evaluate the ML model and to see how well it performs. There are many metrics to consider when evaluating and comparing ML models, in this study, the most important metrics have been selected. The confusion matrix is a useful tool to evaluate and calculate

these model metrics. An example of a confusion matrix is given in table 3.9. True positives are flights that have been predicted to be delayed and that are actually delayed, false positives are flights that have been predicted to be delayed but are actually not delayed, false negatives are flights that have been predicted to not be delayed but are actually delayed, and true negatives are flights that are predicted to be not delayed and are not delayed.

Table 3.9: Confusion matrix

		Predicted	
		Delay	No delay
Actual	Delay	True Positive (TP)	False Negative (FN)
	No delay	False Positive (FP)	True Negative (TN)

The first model metric used is accuracy. The accuracy is defined as the fraction or percentage the model predicted correctly in all of the predictions [32]. The equation for the accuracy is shown in equation 3.5 [32]. Accuracy can be a deceiving evaluation metric in imbalanced datasets [32]. A poorly performing model can have moderate or even high accuracy, and a properly performing model can have low accuracy [32]. This is the reason to combine multiple ML model evaluation metrics to have a better understanding of the performance of the models.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \cdot 100\% \tag{3.5}$$

Misclassification rate or misclassification is directly linked with the accuracy, as it is the difference between 100% and the accuracy. As accuracy is defined as the percentage the model predicted correctly, the misclassification rate is defined as the percentage the model predicted incorrectly [32]. The formula for misclassification can be found in 3.6 [32].

$$Misclassification = \frac{(FP + FN)}{(TP + TN + FP + FN)} \cdot 100\% = 1 - Accuracy \tag{3.6}$$

Recall is defined as the percentage of actual positives or actual delays that were predicted correctly. Recall says something about the model's ability to identify actual positives correctly [32]. The formula for recall is given in equation 3.7. In the flight delay study, a high recall means the model is capable of predicting actual flight delays properly, so in this study a high recall is desirable. Precision on the other hand is defined as the percentage of positive predictions that were actually correct [32]. The formula for precision is given in equation 3.8 [32].

$$Recall = \frac{(TP)}{(TP + FN)} \cdot 100\% \tag{3.7}$$

$$Precision = \frac{(TP)}{(TP + FP)} \cdot 100\% \tag{3.8}$$

The following metric is the F1 score. The F1 score combines both the precision and the recall in a single metric using the harmonic mean between the two metrics [32]. As mentioned previously accuracy can be a deceiving model evaluation metric in imbalanced datasets, this is where the F1 score comes in useful as it gives a better insight into model performance in many cases compared to accuracy. The formula to calculate the F1 score is given in equation 3.9 [32].

$$F1\ score = \frac{(2 \cdot precision \cdot recall)}{(precision + recall)} \cdot 100\% \tag{3.9}$$

Specificity, which is also known as the true negative rate, says something about the models' ability to predict true negatives [32]. In the context of flight delay prediction, this translates into the ability of the model to predict if a given flight has no delay. The formula to calculate specificity is given in equation 3.10.

$$Specificity = \frac{TN}{(TN + FP)} \cdot 100\% \tag{3.10}$$

The next model metric is the area under the receiver operating characteristics (ROC) curve. Figure 3.6 shows an example of a ROC curve [32]. The ROC curve is an evaluation metric used in binary classification problems, it is a probability curve that plots the true positive rate against the false positive rate [32]. The area under the ROC curve (ROC AUC) gives the performance of the model in distinguishing between positive and negative classes, therefore the higher this metric the better performing the model. Calculating this metric is done using built-in tools of Spark and H2O and the ROC curve for each model will be visually compared to find the best-performing model.

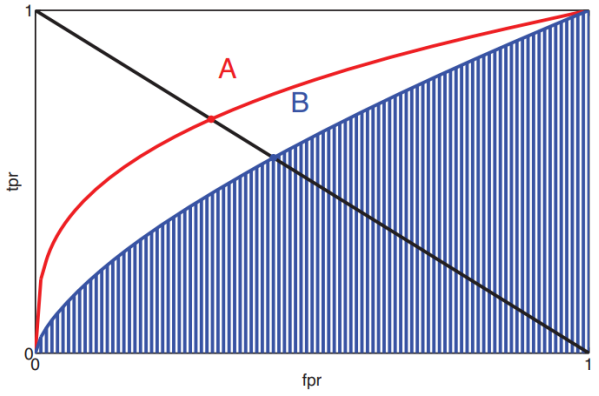

Figure 3.6: ROC AUC curve example [32]

The final model metric is the area under the precision-recall curve, better known as PR AUC. Figure 3.7 shows an example of a PR curve [32]. The PR curve is an evaluation metric used in binary classification problems, it is a probability curve that plots the precision against the recall [32]. The area under the PR curve gives the performance of the model and an area closer to 1 indicates a better-performing model. Calculating this metric is done using built-in tools of Spark and H2O and the PR curve for each model will be visually compared to find the best-performing model.

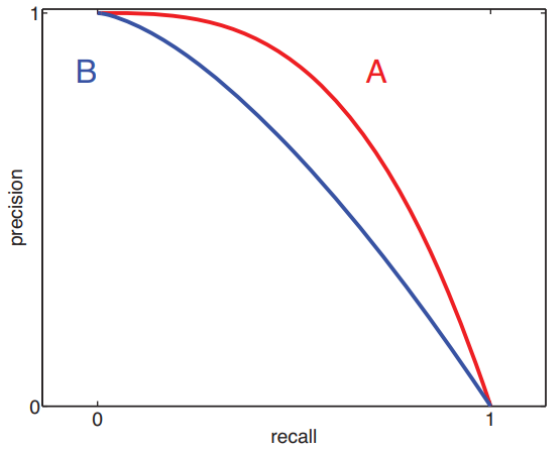

Figure 3.7: PR AUC curve example [32]

This concludes the metrics used to evaluate the model performance using the testing data. The most important model metrics are the ROC AUC and the PR AUC because these metrics give a proper understanding of the model performance in binary classification. Besides the ROC AUC and PR AUC, the F1 score is an important metric as it combines both precision and recall in a single metric. Both precision and recall are important since it is important to detect flight delays and distinguish delayed flights from non-delayed flights. And the last metric is accuracy since this gives an insight into the total percentage of properly performed predictions, however, it is important to use the accuracy in combination with the first three mentioned metrics, as it could be deceiving with imbalanced datasets, such as the flight dataset.

3.3 Tools

This section will give an overview of the two main tools or frameworks used in order to process the data efficiently, as well as to train the models efficiently. These tools are Apache Spark and H2O.

3.3.1 Apache Spark

Apache Spark is an open-source big data analytics and ML tool [33]. Spark supports 4 major programming languages in order to interact with the engine, which are R, Python, Java, and Scala. Since in the scope of this study, the main programming language is set to R, this suits well. Spark has advantages that are useful in this study, including data processing and ML capabilities. The main library in R that is used to interact with Spark is the library Sparklyr [34]. In comma-separated values (CSV) format the flight dataset has a size of 1.1 GB and this is very difficult to process using base R since it requires much higher hardware and

memory specifications. Spark manages this data well and can easily handle and manipulate multiple rows. Regarding data processing Apache Spark is used both for the data processing steps shown in figure 3.2 as well as the train test split shown in figure 3.4. Besides the data processing steps Apache Spark is also used to train two of the ML models summarised in table 3.8, the logistic regression, and the random forest model.

3.3.2 H2O

H2O is an open-source ML framework for big data analysis developed by the h2o.ai company [35]. H2O offers a variety of ML and deep learning tools, such as the gradient boosting machine as well as the feed-forward NN that will be explored in this study. H2O does not offer any efficient way of processing big data such as the flight's dataset, which means that for both the models trained using Spark and H2O, all of the preprocessing will be done through the Spark framework. After processing the data through Spark, the data will be copied inside the H2O framework and will be used to create the train validation test split as outlined in figure 3.5. Subsequently, the gradient boosting machine and feed-forward NN will be tuned, trained, cross-validated, and evaluated based on model performance.

Chapter 4

Results

This chapter consists of the results that followed based on the steps described in the methodology. First, the results from the data processing pipeline will be discussed, this is followed by the discussion of the hyperparameter tuning of each model. This is followed by the results of the final tuned models on the test data. And finally, this chapter is concluded with a comparison of each model, with a final model being picked with the best performance.

4.1 Data processing pipeline

As per the definition of the BTS, a flight is considered delayed if the delay is equal to or more than 15 minutes [17]. Applying this definition to the raw data, the number of delayed and non-delayed arrival flights can be calculated to see the balance of the classes of flight delays. Figure 4.1 shows the imbalance in the arrival delays in the raw data. The raw data is highly imbalanced due to the fact that only 18 % of all flights are delayed. This data, which has 5665109 rows, is used as the input to the data processing pipeline shown in figure 3.2.

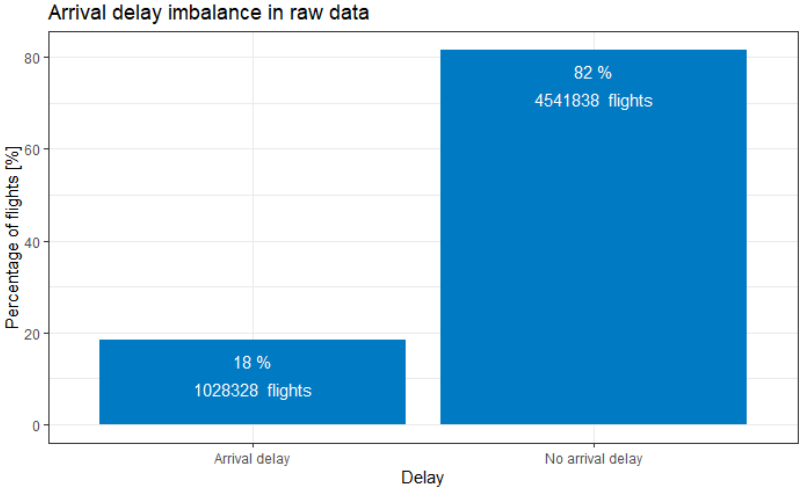

Figure 4.1: Imbalance in arrival flight delays

After processing the data in the pipeline and joining the weather data and the airport congestion data there are only 542421 rows left, which comes down to only 9.57% of the original data! As shown previously in figure 4.1 the arrival delays in the raw data were highly imbalanced. When taking a look at the imbalance in the processed data, it is clear that it is still highly imbalanced. The imbalance in the arrival flight delays of the processed data is shown in figure 4.2.

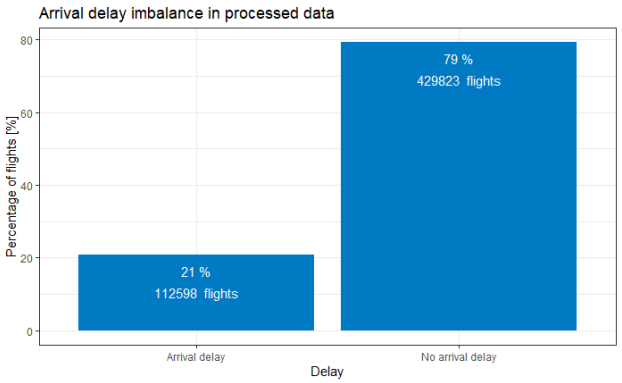

Figure 4.2: Imbalance in arrival flight delays

Figure 4.2 shows that only 21% of the flights have an arrival delay and a big majority of 79 % have no arrival delay. For ML, this forms a problem as this may impact the ability of the model to predict the minority class, which in this case is flights that are delayed. This in turn might impact the accuracy of the trained machine-learning models in a negative way. The imbalance will be solved by performing undersampling during the train test split for spark, which is shown in figure 3.4 in the methodology and train validation test split, which is shown in figure 3.5 in the methodology. By undersampling the train set, random rows of the majority class, which in this case are the non-delayed flights, are removed. For the models trained using spark, 90% of the processed data goes into the training data, and 10% goes into the testing data. The training set is undersampled making the number of delayed flights equal to the number of non-delayed flights. The train test composition after undersampling the training data is shown in figure 4.3. The training set consists of 202862 rows, 50% of which are delayed and 50% of which are non-delayed. The testing set consists of 54198 rows, 21% of which are delayed and 79% of which are non-delayed.

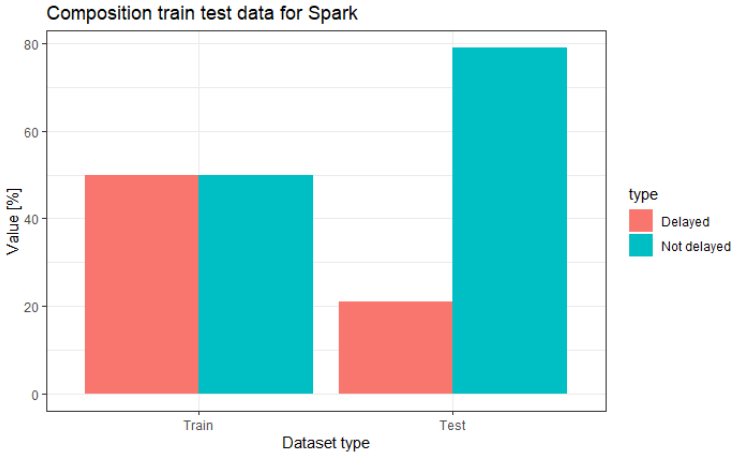

Figure 4.3: Train test split Spark

For the models trained using H2O, 80% of the processed data goes into the training data, 10% goes into the validation data used in cross-validation, and 10% goes into the testing data used for model evaluation. The training data in H2O is undersampled as well, making the number of delayed flights equal to the number of non-delayed flights. The results of the undersampling in the training data for the H2O framework are shown in figure 4.4. The train set consists of 180582 rows, 50% of which are delayed and 50% of which are non-delayed. The validation set consists of 54294 rows, 21% of which are delayed and 79% of which are non-delayed. And finally the test set consist of 54051 rows, 21% of which are delayed and 79% of which are non-delayed.

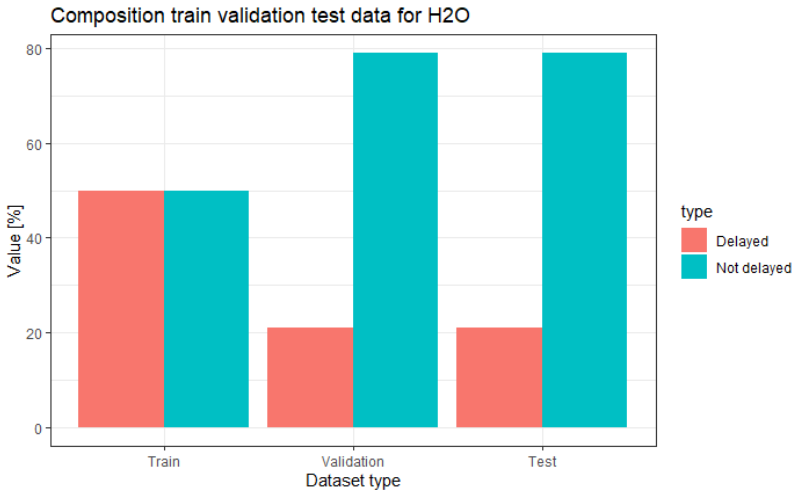

Figure 4.4: Train validation test split H2O

It is shown that in the train set for both Spark as well as for H2O, the number of delayed flights balanced with the non-delayed flights, but keeping the same imbalance in figure 4.1 for the validation and testing datasets. The data used to train the models consist of 36 variables, 35 of which are features, and the last one is the target variable. All the variables used in the final processed dataset can be found in table A.1 in appendix A.

4.2 Hyperparameter tuning models

Each model will be hyperparameter tuned in order to optimize its performance. Hyperparameter tuning is the act of training several different models with several different parameters and gives for each combination of parameters the model performance. After finding the optimal values for the model parameters, a final model can be trained using these parameters with cross-validation. This section will discuss the best combination of parameters for each model.

4.2.1 Logistic regression

As described in the chapter on the methodology the two main parameters to tune in logistic regression are the elastic net regularization denoted as λ, and the regularization denoted as α. Both λ and α have a range between 0 and 1, therefore various values between 0 and 1 are chosen in order to find the optimum parameters

for the model. The values which will be used in hyperparameter tuning are given in table 4.1. For each model trained in hyperparameter tuning the accuracy is calculated, and the parameters of the model with the highest accuracy are picked to train as the final model.

Table 4.1: Logistic regression hyperparameter ranges

Parameter	Symbol	Values
Elastic net regularization	λ	0, 0.25, 0.5, 0.75, 1
Regularization	α	0, 0.25, 0.5, 0.75, 1

The results for the hyperparameter tuning for the logistic regression model are summarised in table 4.2. Using an elastic net regularization or λ and a regularization or α value of 0 gives the best performance in hyperparameter tuning. These parameters will be used to train the final model.

Table 4.2: Logistic regression optimal parameters

Parameter	Symbol	Optimal value
Elastic net regularization	λ	0
Regularization	α	0

4.2.2 Random forest

As described in the chapter on the methodology the two significant parameters to tune in random forest models are the maximum depth of the tree and the number of trees. In Spark, the maximum depth of the tree is also denoted as max_depth, and the number of trees is denoted as num_trees. In order to find the optimal value a wide enough range of values is used without losing too much time in the computation of the hyperparameter tuning. The values which will be used in hyperparameter tuning are given in table 4.3. For each model trained in hyperparameter tuning the accuracy is calculated, and the parameters of the model with the highest accuracy are picked to train as the final model.

Table 4.3: Random forest hyperparameter ranges

Parameter	Symbol in Spark	Values
Maximum depth	max_depth	1, 3, 5, 7, 10
Number of trees	num_trees	1, 3, 5, 7, 10, 25, 50

The results for the hyperparameter tuning for the random forest model are summarised in table 4.4. Using a maximum depth of 10 and a number of trees of 50 gives the best performance in hyper parameter tuning. These parameters will be used to train the final model.

4.2.3 Gradient boosting machine

As described in the chapter on the methodology there are several significant parameters to tune in gradient boosting machines using H2O. The main metrics tuned in hyperparameter tuning are the maximum depth,

Table 4.4: Random forest optimal parameters

Parameter	Symbol in Spark	Values
Maximum depth	max_depth	10
Number of trees	num_trees	50

the row sampling rate per tree, the column sampling rate per tree, the column sampling rate, the minimum rows to sample, the number of bins for continuous variables, the number of bins for categorical variables, relative error improvement threshold, for three different histogram types: UniformAdaptive, QuantilesGlobal, and RoundRobin. In order to find the optimal value a wide enough range of values without losing too much time in the computation of the hyperparameter tuning. The values which will be used in hyperparameter tuning are given in table 4.5. For each model trained in hyperparameter tuning the accuracy is calculated, and the parameters of the model with the highest accuracy are picked to train as the final model.

Table 4.5: Gradient boosting machine hyperparameter ranges

Parameter	Symbol in H2O	Values
Maximum depth	max_depth	1, 3, 5, 7, 9, 11, 13, 15 .. 29
Row sampling rate	sample_rate	0.20, 0.21, 0.22, 0.23, .. 1.00
Column sampling rate	col_sample_rate	0.20, 0.21, 0.22, 0.23, .. 1.00
Column sample rate per tree	col_sample_rate_change_per_level	0.90, 0.91 0.92, 0.93, .. 1.10
Minimum observations per leaf	min_rows	1, 2, 4, 8, 16, 32, 64, .. 2048
Bins for continuous features	nbins	16, 32, 64, 128, 256, .. 1024
Bins for categorical features	nbins	16, 32, 64, 128 .. 4096
Minimum error improvement	min_split_improvement	0, 10^{-8}, 10^{-6}, 10^{-4}

The results for the hyperparameter tuning of the gradient boosting machine model are summarised in table 4.6. Using these parameters with a histogram type of RoundRobin gives the best model in parameter tuning. These parameters will be used to train the final model.

Table 4.6: Gradient boosting machine optimal parameters

Parameter	Symbol in H2O	Values
Maximum depth	max_depth	17
Row sampling rate	sample_rate	0.91
Column sampling rate	col_sample_rate	0.33
Column sample rate per tree	col_sample_rate_change_per_level	0.95
Minimum observations per leaf	min_rows	16
Bins for continuous features	nbins	512
Bins for categorical features	nbins	64
Minimum error improvement	min_split_improvement	0

4.2.4 Feed-forward neural network

As described in the chapter on the methodology there are several significant parameters to tune in feed-forward NNs using H2O. The main metrics tuned in hyperparameter tuning are the number of hidden layers and the number of nodes in each hidden layer, the input dropout ratio, the learning rate, and the learning rate annealing. In order to find the optimal value a range of values is used without losing too much time in the computation of the hyperparameter tuning. The values which will be used in hyperparameter tuning are given in table 4.7. For each model trained in hyperparameter tuning the accuracy is calculated. For each model trained in hyperparameter tuning the accuracy is calculated, and the parameters of the model with the highest accuracy are picked to train as the final model.

Table 4.7: Feed-forward neural network hyperparameter ranges

Parameter	**Symbol in H2O**	**Values**
Hidden layers and nodes	hidden	$32 \rightarrow 32 \rightarrow 32, 64 \rightarrow 64, 100 \rightarrow 100 \rightarrow 100$
Input dropout ratio	input_dropout_ratio	0, 0.05
Learning rate	rate	0.01, 0.02
Learning rate annealing	rate_annealing	10^{-8},10^{-7},10^{-6}

The results for the hyperparameter for the feed-forward NN tuning are summarised in table 4.8. Hyperparameter tuning shows that two hidden layers with 64 nodes each give the best results. These parameters will be used to train the final model.

Table 4.8: Feed-forward neural network optimal parameters

Parameter	**Symbol in H2O**	**Values**
Hidden layers and nodes	hidden	$64 \rightarrow 64$
Input dropout ratio	input_dropout_ratio	0.05
Learning rate	rate	0.02
Learning rate annealing	rate_annealing	10^{-6}

4.3 Final model results

Using the results of hyperparameter tuning, the optimal parameter values for each model, the final machine learning models will be trained. This section will describe the confusion matrix obtained using the test set predictions on the final model. The confusion matrix is then used to calculate for each model the respective model metrics that are explained in the methodology section and the performance will be elaborated.

4.3.1 Logistic regression

The confusion matrix for predictions using the test dataset with the final model for logistic regression is shown in table 4.9. The final model uses the optimal parameters summarised in table 4.2. The confusion matrix is used to calculate the model evaluation metrics.

Table 4.9: Confusion matrix for logistic regression

		Predicted	
		Delay	No delay
Actual	Delay	7137	4143
	No delay	15040	28025

Using the data in the confusion matrix in table 4.9 the accuracy is calculated to be 65%. The misclassification rate is derived from the accuracy and is 35%. The calculation is shown in equations 4.1.

$$\begin{aligned} Accuracy &= \frac{(7137 + 28025)}{(7137 + 15040 + 28025 + 4143)} \cdot 100\% \approx 65\% \\ Misclassification &= 100\% - accuracy \approx 35\% \end{aligned} \tag{4.1}$$

The recall is calculated to be 63% and the precision is calculated to be 32%. The F1-score is derived from the precision and recall and is calculated at 42%. The specificity is calculated at 65%. The calculation of the precision, recall, F1-score, and specificity is shown in equation 4.4.

$$\begin{aligned} Recall &= \frac{7137}{(7137 + 4143)} \cdot 100\% \approx 63\% \\ Precision &= \frac{7137}{(7137 + 15040)} \cdot 100\% \approx 32\% \\ F1\ score &= \frac{(2 \cdot precision \cdot recall)}{(precision + recall)} \cdot 100\% \approx 42\% \\ Specificity &= \frac{28025}{(28025 + 15040)} \cdot 100\% \approx 65\% \end{aligned} \tag{4.2}$$

The ROC curve is plotted in the comparison between all ROC curves in figure 4.10, and the PR curve is plotted in the comparison between all PR curves in figure 4.11. These curves are more important when comparing models, when looking at the performance of individual models, the area under these curves is more important. The built-in calculator of Spark is used to calculate the area under these curves which are important performance metrics. The area under the ROC curve is equal to 0.69 or 69% and the area under the PR curve is equal to 0.38 or 38%. An overview of the model evaluation metrics for logistic regression is shown in figure 4.5.

When looking at the performance metrics of the logistic regression model in figure 4.5 it shows balanced metrics that range between 63% and 69% in value for the accuracy, recall, ROC AUC, and specificity. The model seems to perform not so well with regards to precision, PR AUC, and the F1 score, with values ranging from 32% to 42%, these lower values can be the result of imbalanced data and dataset size.

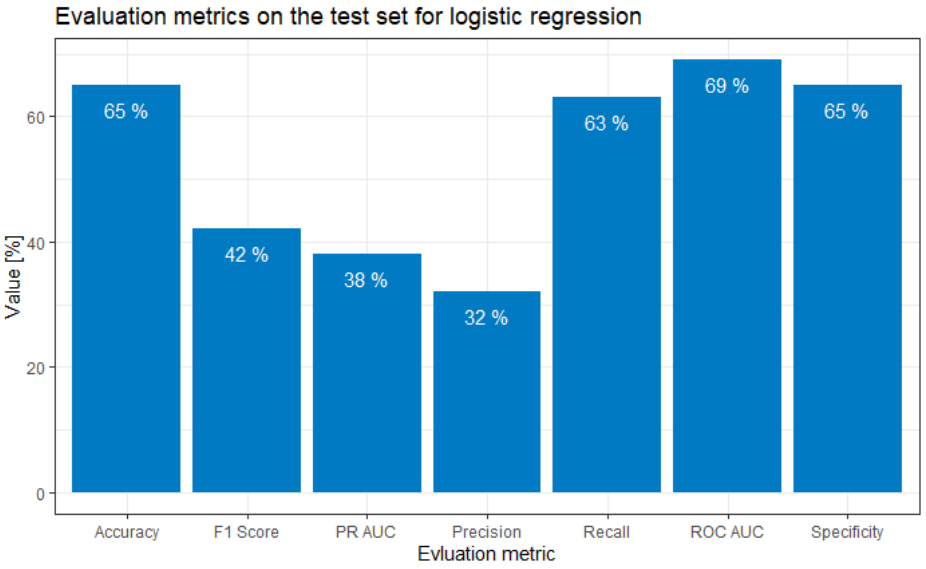

Figure 4.5: Evaluation metrics for logistic regression

4.3.2 Random forest

The confusion matrix for predictions using the test dataset with the final model for the random forest algorithm is shown in table 4.10. The final model uses the optimal parameters summarised in table 4.4. The confusion matrix is used to calculate the model evaluation metrics.

Table 4.10: Confusion matrix for random forest

		Predicted	
		Delay	No delay
Actual	Delay	7428	3852
	No delay	12626	30439

Using the data in the confusion matrix in table 4.10 the accuracy is calculated to be 70%. The misclassification rate is derived from the accuracy and is 30%. The calculation is shown in equations 4.3.

$$\begin{aligned} Accuracy &= \frac{(7428 + 30439)}{(7428 + 12626 + 30439 + 3852)} \cdot 100\% \approx 70\% \\ Misclassification &= 100\% - accuracy \approx 30\% \end{aligned} \tag{4.3}$$

The recall is calculated to be 66% and the precision is calculated to be 37%. The F1-score is derived from the precision and recall and is calculated at 47%. The specificity is calculated at 71%. The calculation of the precision, recall, F1-score, and specificity is shown in equation 4.4.

$$
\begin{aligned}
Recall &= \frac{7428}{(7428 + 3852)} \cdot 100\% \approx 66\% \\
Precision &= \frac{7428}{(7428 + 12626)} \cdot 100\% \approx 37\% \\
F1\ score &= \frac{(2 \cdot precision \cdot recall)}{(precision + recall)} \cdot 100\% \approx 47\% \\
Specificity &= \frac{30439}{(30439 + 12626)} \cdot 100\% \approx 71\%
\end{aligned}
\tag{4.4}
$$

The ROC AUC and PR AUC is calculated using the built-in calculator of Spark. The area under the ROC curve is equal to 0.75 or 75% and the area under the PR curve is equal to 0.48 or 48%. An overview of the model evaluation metrics for the random forest model is shown in figure 4.6.

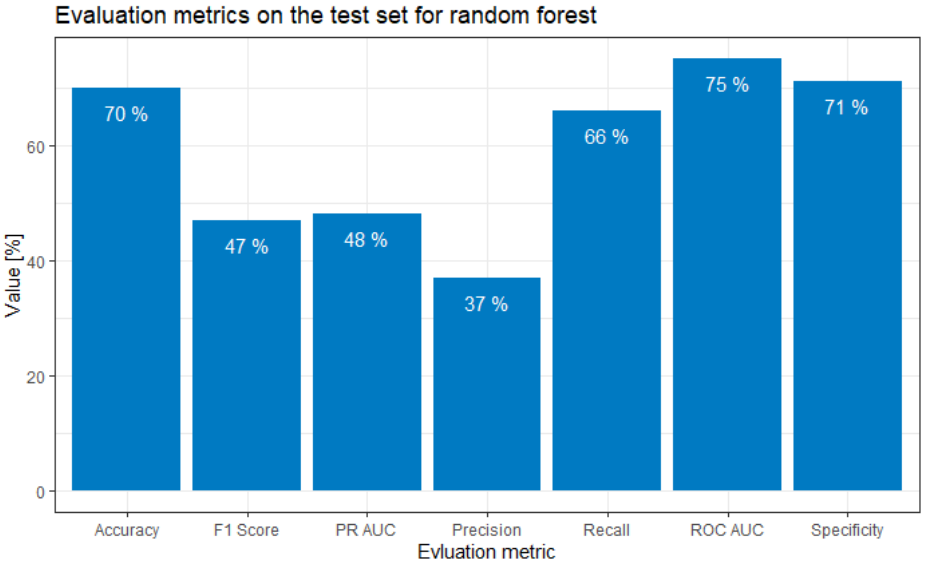

Figure 4.6: Evaluation metrics for random forest

When looking at the performance metrics of the random forest model in figure 4.6 it shows balanced metrics that range between 66% and 75% in value, for the accuracy, recall, ROC AUC, and specificity. The model seems to perform poorer in the F1 score, PR AUC, and precision which can be due to the fact that the dataset is imbalanced and due to the size.

4.3.3 Gradient boosting machine

The confusion matrix for predictions using the test dataset with the final model for the gradient boosting machine algorithm is shown in table 4.11. The final model uses the optimal parameters summarised in table 4.6. The confusion matrix is used to calculate the model evaluation metrics.

Table 4.11: Confusion matrix for gradient boosting machine

		Predicted	
		Delay	No delay
Actual	Delay	9901	1350
	No delay	12152	31019

Using the data in the confusion matrix in table 4.11 the accuracy is calculated to be 75%. The misclassification rate is derived from the accuracy and is 25%. The calculation is shown in equations 4.5.

$$\begin{aligned} Accuracy &= \frac{(9901 + 31019)}{(9901 + 12152 + 31019 + 1350)} \cdot 100\% \approx 75\% \\ Misclassification &= 100\% - accuracy \approx 25\% \end{aligned} \tag{4.5}$$

The recall is calculated to be 88% and the precision is calculated to be 45%. The F1-score is derived from the precision and recall and is calculated at 60%. The specificity is calculated at 72%. The calculation of the precision, recall, F1-score, and specificity is shown in equation 4.6.

$$\begin{aligned} Recall &= \frac{9901}{(9901 + 1350)} \cdot 100\% \approx 88\% \\ Precision &= \frac{9901}{(9901 + 12152)} \cdot 100\% \approx 45\% \\ F1\ score &= \frac{(2 \cdot precision \cdot recall)}{(precision + recall)} \cdot 100\% \approx 60\% \\ Specificity &= \frac{31019}{(31019 + 12152)} \cdot 100\% \approx 72\% \end{aligned} \tag{4.6}$$

Both the area under the ROC curve and the area under the PR curve can be calculated using the built-in calculator in H2O. The area under the ROC curve is equal to 0.89 or 89% and the area under the PR curve is equal to 0.68 or 68%. An overview of the model evaluation metrics for the gradient boosting machine model is shown in figure 4.7.

When looking at the performance metrics of the gradient boosting machine model in figure 4.7 it shows much better metrics some of them in the range between 60% and 89% in value, the better performing metrics are the accuracy, the F1 score, the PR AUC, the recall, the ROC AUC, and the specificity. There seems to be a pattern in a poor value of precision, similarly to the previous models, which can be the result of a imbalanced dataset. It shows a high value for the area under the ROC and PR curve, which indicates a good model in binary classification.

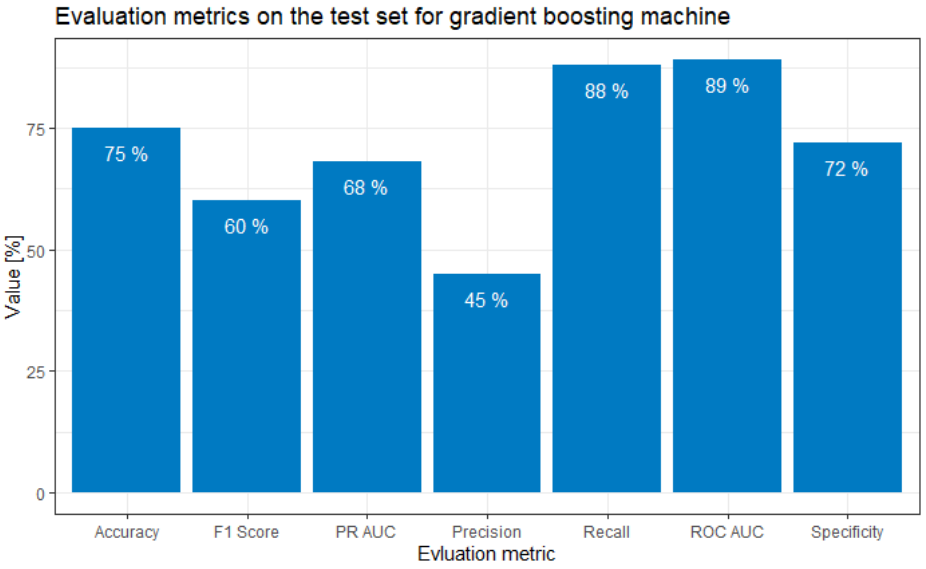

Figure 4.7: Evaluation metrics for gradient boosting machine

4.3.4 Feed-forward neural network

The confusion matrix for predictions using the test dataset with the final model for the feed-forward neural network algorithm is shown in table 4.12. The final model uses the optimal parameters summarised in table 4.8. The confusion matrix is used to calculate the model evaluation metrics.

Table 4.12: Confusion matrix for feed-forward neural network

		Predicted	
		Delay	No delay
Actual	Delay	9842	1409
	No delay	27407	15764

Using the data in the confusion matrix in table 4.12 the accuracy is calculated to be 47%. The misclassification rate is derived from the accuracy and is 53%. The calculation is shown in equations 4.7.

$$\begin{aligned} Accuracy &= \frac{(9842 + 15764)}{(9842 + 27407 + 15764 + 1409)} \cdot 100\% \approx 47\% \\ Misclassification &= 100\% - accuracy \approx 53\% \end{aligned} \tag{4.7}$$

The recall is calculated to be 87% and the precision is calculated to be 26%. The F1-score is derived from the precision and recall and is calculated at 40%. The specificity is calculated at 37%. The calculation of the precision, recall, F1-score, and specificity is shown in equation 4.8.

$$
\begin{aligned}
Recall &= \frac{9842}{(9842 + 1409)} \cdot 100\% \approx 87\% \\
Precision &= \frac{9842}{(9842 + 27407)} \cdot 100\% \approx 26\% \\
F1\ score &= \frac{(2 \cdot precision \cdot recall)}{(precision + recall)} \cdot 100\% \approx 40\% \\
Specificity &= \frac{15764}{(15764 + 27407)} \cdot 100\% \approx 37\%
\end{aligned}
\tag{4.8}
$$

Both the area under the ROC curve and the area under the PR curve can be calculated using the built-in calculator in H2O. The area under the ROC curve is equal to 0.73 or 73% and the area under the PR curve is equal to 0.43 or 43%. An overview of the model evaluation metrics for the feed-forward neural network model is shown in figure 4.8. When looking at the performance metrics of the feed-forward neural network model in figure 4.8 it shows quite imbalanced performance metrics that range between 26% for the lowest metric and 87% for the highest metric in value. One of the big outliers is the very good recall performance of 87% and the decent performance of ROC AUC, compared to very poor performance of the remaining metrics, indicating a good performance in detecting positives with a high recall but a poor performance overall. This also means that this type of tabular data from flights and weather does not work very well with feed-forward neural networks.

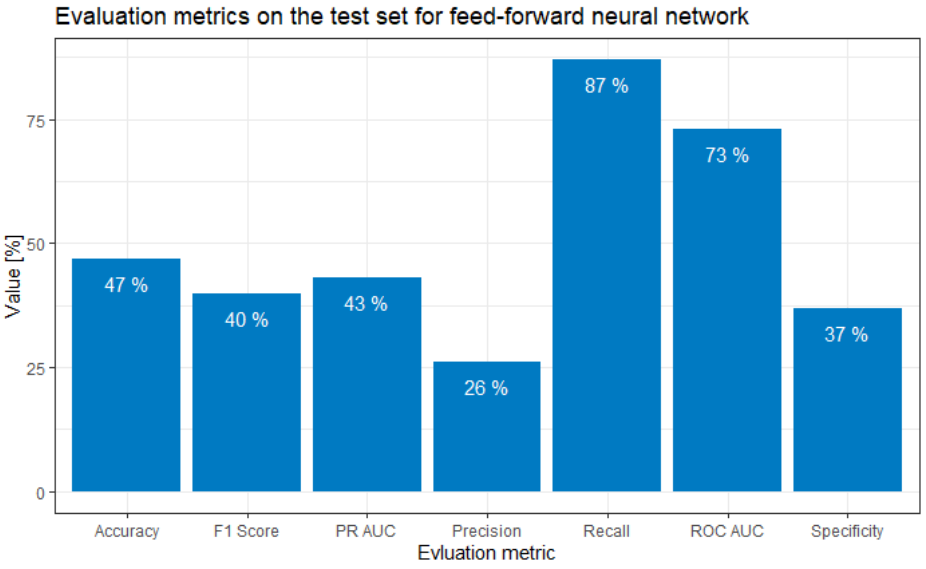

Figure 4.8: Evaluation metrics for feed-forward neural network

4.4 Final model comparison

Now that all the evaluation metrics have been obtained from the final tuned models using predictions on the test data, the next step is to compare each model and pick the best-performing model as the final solution to the delay prediction problem. Figure 4.9 shows the comparison of the accuracy, F1 score, specificity, area under the ROC curve, and the area under the PR curve. Since the F1 score is a metric that takes both

the precision and the recall into account, it suffices to look at the F1 score when comparing models. When looking at the visual comparison in figure 4.9 of the important evaluation metrics it is clear that the gradient boosting machine has the best performance among the four machine learning models. Gradient boosting machine wins when looking at accuracy, the F1 score, PR AUC, and ROC AUC, and beats the random forest slightly in terms of specificity.

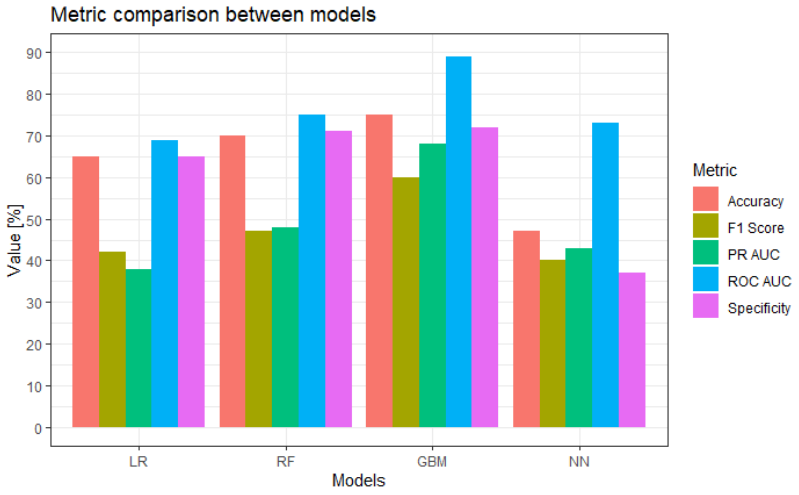

Figure 4.9: Comparison of evaluation metrics between the different models:
Logistic regression (LR), Random Forest (RF)
Gradient Boosting Machine (GBM)
Feed-Forward Neural Network (NN)

Also when looking at a comparison of the ROC curves of each model in figure 4.10, the gradient boosting machine clearly has a higher area under the Receiver Operating Characteristics curve. This indicates a better performance as a binary classification model. Within ROC AUC performance the gradient boosting machine is followed by the random forest model, the feed-forward neural network, and lastly the logistic regression model.

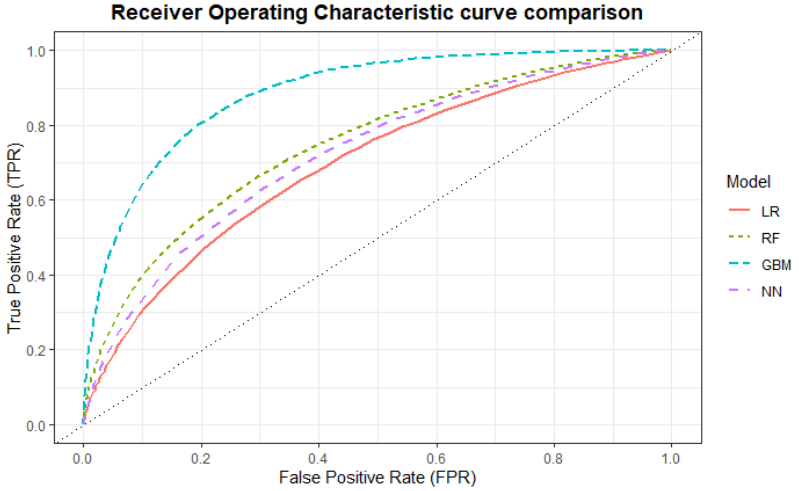

Figure 4.10: Comparison of the ROC curves between the different models:
Logistic regression (LR), Random Forest (RF)
Gradient Boosting Machine (GBM)
Feed-Forward Neural Network (NN)

A comparison of the precision-recall curves of all models is given in figure 4.11. From the precision-recall curve, it is clearly visible that the gradient boosting machine has the best performance among all models in terms of PR AUC. Within PR AUC performance the gradient boosting machine is again followed by the random forest model, the feed-forward neural network, and lastly the logistic regression model.

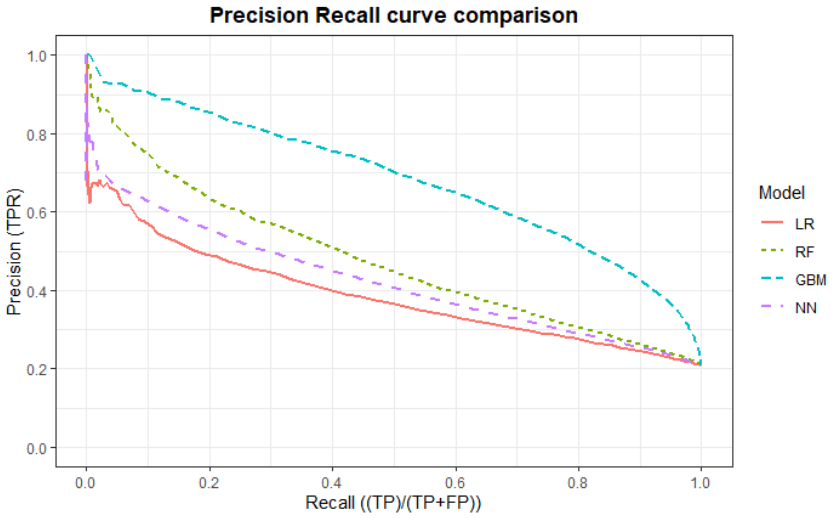

Figure 4.11: Comparison of the PR curves between the different models:
Logistic regression (LR), Random Forest (RF)
Gradient Boosting Machine (GBM)
Feed-Forward Neural Network (NN)

In specificity, the random forest model performs similarly to the gradient boosting machine, but in all other metrics, the gradient boosting machine has the best performance. Random forest is the second-best predictive model among the four. The logistic regression model is a very close third among the models, outperforming the feed-forward neural network in accuracy, ROC AUC, and specificity. Of the four models, the feed-forward neural network has the poorest performance. Besides having a lower accuracy, the model especially performs very poorly in specificity compared to the other models, where the other models have slightly more balanced metrics. It is interesting however to note that the feed-forward neural network has a much higher recall, which is visible in figure 4.8 compared to the other three models. The outliers in the specificity and recall indicate that this type of machine learning model is not suitable for tabular data such as the flight and weather data in table A.1. Also, it is important to note that all models perform poorly in precision. This poor performance in precision is also reflected in the values for the F1 score since the F1 score is directly derived from precision and recall as well. The poor performance in precision, is likely due to the highly imbalanced nature of the dataset as well as its size, with only a minority of the dataset being delayed and a large majority being non-delays. Out of all four models, the gradient boosting machine is the best model with the best performance in all categories and therefore the model most suitable to perform flight delay predictions. This model seems to beat the lightGBM model, which has a similar architecture in Miguel Lambelho et al [4], on the following evaluation metrics, F1 score with an 8% improvement, recall with a 36% improvement, ROC AUC with a 10% improvement.

Chapter 5

Environmental and social implications

As mentioned at the very beginning of this report, in the introduction, flight delays bring a significant negative impact on people, the economy, and the environment. In this chapter, these implications, and how the solution proposed can help to reduce these implications will be assessed and discussed.

5.1 Negative impact of flight delays

Flight delays can be extremely troublesome for travelers, especially if they have a layover flight. If the time window between the two flights is very narrow and the initial flight experiences a delay, this could result in some passengers missing the second flight. Missing the second flight might cost a lot of money for the passengers if they booked flights on two different carriers, especially for traveling families. In the case of two different carriers, there would most likely not be a refund. This would result in re-booking multiple flights, which can cost a lot of money, plus the discomfort and stress during the journey. So without a doubt, flight delays have their share in the social implications caused by air travel. Besides the social implications caused by flight delays, there are also considerably big economic implications. A study from 2013 [2] shows that a decrease of just 10 percent in flight delays would increase the United States' net welfare by $17.6 billion and a decrease of 30 percent would increase the United States' net welfare by a staggering $38.5 billion. Besides the economic impact, flight delays have also a severe impact on the environment. A study performed in 2018 [3] shows that in the year 2017 the emission due to flight delays were estimated at 5520 tonnes while excess fuel usage was estimated at 1.752.937 liters. It is important to note that these are the excess emissions and fuel usage from just departing or arriving later than scheduled. Some of these negative implications of flight delays can be addressed by using machine learning, which will be discussed in the next section. The impact or decrease of these social, environmental, and economic impacts of flight delays by deploying a machine learning model, similar to the solution proposed in this study will be discussed in the next section.

5.2 Impact of the solution

The solution, based on the gradient boosting machine algorithm, proposed in the previous chapter is capable of detecting flight delays with an accuracy of 75% and an area under the curve of 89% for the ROC curve. This means that the proposed model can be used to detect flight delays in most cases with data known before the flight has taken off. The proposed machine learning model can be used as a supportive tool in airport management and flight scheduling. During the scheduling of the flight, the proposed solution can help anticipate flight delays or the number of delayed flights in the top 10 airports in the United States airport network. In airport management, a machine learning algorithm like the solution proposed in the previous chapter can support strategic decision-making, anticipating flight delays even before the flight has taken off. It is important to note that with an accuracy of 75% and an area under the ROC curve of 89%, the model can not be deployed in a critical assessing situation, since there will still be some flights incorrectly classified. However, as mentioned in the previous section, a decrease of just 10% in flight delays would already increase the United States' net welfare by a staggering $17.6 billion. This means that even without an ideal and perfect response such a model in production could already decrease many of the implications that flight delays cause.

Chapter 6

Conclusions

This section presents the conclusions of the study on flight delay prediction in the US airport network. First, a recapitulation is given of the results obtained in this study, this is followed by the limitations of the study and concluded with recommendations for further work to conclude the study.

6.1 Recapitulation

This section contains a recapitulation of the results. Initially, the process started by selecting suitable data, which was chosen to be flight data originating from the Bureau of Transport statistics [7] and weather data origination from National Oceanic and Atmospheric Administration [8].

This was followed by processing the data, which included processing the flight data, merging this with the weather data and airport congestion, applying one-hot encoding, and removing any rows with NA values, as shown in figure 3.2 and 3.3 in the methodology. This data processing resulted in a dataset with 542.421 rows, from which 21% were delayed on arrival and 79% were not delayed on arrival as shown in the bar chart in figure 4.2. The processed dataset was then split in the train-test and train-validation-test split as shown in figure 3.4 and figure 3.5 in the methodology. During this process, the train data was undersampled to achieve a balanced training set to train the machine learning models, the resulting train-test and train-validation-test data composition are shown in figure 4.3 and figure 4.4.

After obtaining the train-test and train-validation-test data, the four chosen models: Logistic Regression (LR), Random Forest (RF), Gradient Boosting Machine (GBM), and Feed-Forward Neural Network (NN) were each tuned for their respective parameters. The optimal values of their respective parameters are summarised in tables 4.2, 4.4, 4.6, and 4.8. The optimal parameter values obtained through parameter tuning were then used to train the final models for each algorithm. The final models were then evaluated using the test data and were each compared by different evaluation metrics in figure 4.9 for accuracy, F1 score, PR AUC, ROC AUC, and specificity, a recapitulation of this comparison is given in table 6.1. From this comparison, it became clear that the Gradient Boosting Machine (GBM) is the best model for each metric. The four different models were also compared with the Receiver Operating Characteristic (ROC) curve in

figure 4.10, in which the Gradient Boosting Machine (GBM) is clearly by far the best-performing model. The four models were also compared with the Precision-Recall (PR) curve in figure 4.11, in which, again, the Gradient Boosting Machine is by far the best-performing model. The conclusion from the model comparison is that the Gradient Boosting machine algorithm has the best capability of predicting arrival flight delays among the four chosen machine learning algorithms.

Table 6.1: Recapitulation of model comparison

Model	Accuracy	F1 Score	PR AUC	ROC AUC	Specificity
LR	65%	42%	38%	69%	65%
RF	70%	47%	48%	75%	71%
GBM	75%	60%	68%	89%	72%
NN	47%	40%	43%	73%	37%

6.2 Limitations

There were several limitations in this study with regard to the data, tools, and applications. One of the limitations is that in this study only the data from 2017 is considered. Limited data grants you limits that can be achieved with model parameter tuning and cross-validation, an increase in data could overcome these limits and achieve better results.

One of the limitations is that the proposed solution outperforms the model with a similar architecture and proper evaluation in Miguel Lambelho et al. [4] on F1 score, recall, and ROC AUC. However, the model proposed by Miguel lambelho et al. outperforms the proposed solution on accuracy and precision, but Miguel lambelho et al. only considers a single airport which is London Heathrow airport, and not multiple airports.

As mentioned in the literature study, in Wei Shao et al. [13] flight trajectory data from a closed database was used. This flight trajectory data is used to model trajectory congestion. As a result of this database not being public, this data could not be used in this study to create and engineer additional features. The flight trajectory congestion combined with the airport congestion could have improved the model performance even further.

Another limitation with regard to the hardware used in this study is the specification of the computer used to process the data and train the machine-learning models. The computer used in this study has only 4 CPU cores and 16 GB of RAM memory. The hardware used limits the speed of running the scripts to process the data, tune the models, and train and compare the models. The markdown script in the GitHub repository [1] used to develop the code used in this study, took more than 10 hours to run with the hardware used. This extremely limits the amount of testing an individual can perform as it is only possible to run a script and perform tests with the full data twice a day. Besides the time it takes to run the script, no other operations are possible on the computer as the script consumes all the possible RAM while running. Using a larger or better computer that possesses more CPU cores and more RAM, can greatly improve the time to develop and engineer features, the time to test said features, and also allows you to use more data. Instead of using

the data of a single year, it could be possible to use the data from multiple years, which as mentioned before could improve the overall performance of each model.

And lastly, this machine learning model and study is limited just to flights in the United States airport network, therefore this cannot be applied to for instance the European airport network or the Asian or South American airport network. A more general applicable machine learning model can be obtained by combining datasets of multiple large airports on multiple continents and using that to train models.

6.3 Further work

For future studies that relate to flight delay prediction in the US airport network, there are several things that could be taken into consideration to achieve better results. My first recommendation is to use much more data. Instead of using data from a single year, it can improve highly to use data from multiple years, 3 years' worth of data, or even 5 to 10 years' worth of data. From the roughly 5.5 million rows that were fed into the data pipeline only 550.000 rows were left, meaning that roughly 90% of the data did not make it into the final data used to train and evaluate the models. Because the data is so imbalanced, more data that is being used, allows you to feed more positives or delays in the models, which in turn might solve the precision issue encountered in this study. The recommendation to use more data comes concurrently with the recommendation to use a computer with better specs. Not only will better specifications allow to use of more data, but they will also allow running the scripts faster, granting more time for feature engineering research and testing using the entire data set. Instead of using a physical computer, it is also possible to go for cloud computing and lease an on-demand Linux server through Amazon Web Services [36]. This Linux server can run RStudio Server and can easily be scaled up and down in the number of processor cores and RAM, and can be paid only per minute of use. This way it is possible to perform long tests with the full data in the cloud while running tests with smaller data samples locally on the computer used to access the cloud server, and will eventually be more convenient during the development of code.

Besides recommendations in data quantity and computation specifications, there are also several points to recommend in future work regarding the approach to the modeling problem and additional features to add. In the study, an approach is done to predict flight delays for the top 10 airports in the US airport network using a single machine learning algorithm. It may be more effective to train a machine learning algorithm for each airport of these top 10 airports and deploy machine learning models for each respective airport individually, an approach similar to what has been seen in the literature study in Peng Hu et al. [9]. There might be less variance and noise in the data and the models might be better able to extract information and have a better performance, as it might work better to use several machine learning models to solve a single problem. It could be good to attempt to obtain access to the flight trajectory database [15] by partnering up with the Federal Aviation Administration of the Department of Transportation in future research. The flight trajectory database can be used to add flight trajectory congestion as a feature to the data to improve model performance further.

References

1. KILIÇ, Kerim. *Flight delay prediction GitHub repository*. 2022. Available also from: `https://github.com/kerim-kilic/flight-delay-prediction`.
2. PETERSON, Everett B; NEELS, Kevin; BARCZI, Nathan; GRAHAM, Thea. The economic cost of airline flight delay. *Journal of Transport Economics and Policy (JTEP)*. 2013, vol. 47, no. 1, pp. 107–121.
3. DISSANAYAKA, DMMS; ADIKARIWATTAGE, V; PASINDU, HR. Evaluation of Emissions from Delayed Departure Flights at Bandaranaike International Airport (BIA). In: *11th Asia Pacific Transportation and the Environment Conference (APTE 2018)*. Atlantis Press, 2019, pp. 183–186.
4. LAMBELHO, Miguel; MITICI, Mihaela; PICKUP, Simon; MARSDEN, Alan. Assessing strategic flight schedules at an airport using machine learning-based flight delay and cancellation predictions. *Journal of Air Transport Management*. 2020, vol. 82, p. 101737.
5. *Eurocontrol: The European Organisation for the Safety of Air Navigation*. 2022. Available also from: `https://www.eurocontrol.int/what-we-offer`.
6. CHOI, Sun; KIM, Young Jin; BRICENO, Simon; MAVRIS, Dimitri. Prediction of weather-induced airline delays based on machine learning algorithms. In: *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 2016, pp. 1–6.
7. *Airline on-time statistics, Bureau of Transport Statistics, United States Department of Transportation*. 2022. Available also from: `https://www.transtats.bts.gov/ONTIME/`.
8. *National Oceanic and Atmospheric Administration: Global Hourly - Integrated Surface Database (ISD)*. 2022. Available also from: `https://www.ncei.noaa.gov/products/land-based-station/integrated-surface-database`.
9. HU, Peng; ZHANG, Jianping; LI, Ning. Research on Flight Delay Prediction Based on Random Forest. In: *2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*. IEEE, 2021, pp. 506–509.
10. *Civil Aviation Administration of China*. 2022. Available also from: `http://www.caac.gov.cn/en/HYYJ/`.
11. YU, Bin; GUO, Zhen; ASIAN, Sobhan; WANG, Huaizhu; CHEN, Gang. Flight delay prediction for commercial air transport: A deep learning approach. *Transportation Research Part E: Logistics and Transportation Review*. 2019, vol. 125, pp. 203–221.
12. KIM, Young Jin; CHOI, Sun; BRICENO, Simon; MAVRIS, Dimitri. A deep learning approach to flight delay prediction. In: *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 2016, pp. 1–6.

13. SHAO, Wei; PRABOWO, Arian; ZHAO, Sichen; TAN, Siyu; KONIUSZ, Piotr; CHAN, Jeffrey; HEI, Xinhong; FEEST, Bradley; SALIM, Flora D. Flight delay prediction using airport situational awareness map. In: *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2019, pp. 432–435.
14. *Weather Underground: Los Angeles International airport*. 2022. Available also from: `https://www.wunderground.com/history/daily/us/ca/los-angeles/KLAX/date`.
15. *System Wide Information Management (SWIM), Federal Aviation Administration, United States Department of Transportation*. 2022. Available also from: `https://www.faa.gov/air_traffic/technology/swim/`.
16. GUI, Guan; LIU, Fan; SUN, Jinlong; YANG, Jie; ZHOU, Ziqi; ZHAO, Dongxu. Flight delay prediction based on aviation big data and machine learning. *IEEE Transactions on Vehicular Technology*. 2019, vol. 69, no. 1, pp. 140–150.
17. *Airline On-Time Performance and Causes of Flight Delays, Bureau of Transport Statistics, United States Department of Transportation*. 2022. Available also from: `https://www.bts.dot.gov/explore-topics-and-geography/topics/airline-time-performance-and-causes-flight-delays`.
18. ZOU, Xiaonan; HU, Yong; TIAN, Zhewen; SHEN, Kaiyuan. Logistic regression model optimization and case analysis. In: *2019 IEEE 7th international conference on computer science and network technology (ICCSNT)*. IEEE, 2019, pp. 135–139.
19. *Spark Machine Learning - Logistic Regression*. 2022. Available also from: `https://rdrr.io/cran/sparklyr/man/ml_logistic_regression.html`.
20. LUO, Hanwu; PAN, Xiubao; WANG, Qingshun; YE, Shasha; QIAN, Ying. Logistic regression and random forest for effective imbalanced classification. In: *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2019, vol. 1, pp. 916–917.
21. DEMIDOVA, Liliya; IVKINA, Mariya. Defining the Ranges Boundaries of the Optimal Parameters Values for the Random Forest Classifier. In: *2019 1st International Conference on Control Systems, Mathematical Modelling, Automation and Energy Efficiency (SUMMA)*. IEEE, 2019, pp. 518–522.
22. *Spark Machine learning - Random forest*. 2022. Available also from: `https://rdrr.io/cran/sparklyr/man/ml_random_forest.html`.
23. HANCOCK, John; KHOSHGOFTAAR, Taghi M. Leveraging lightgbm for categorical big data. In: *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE, 2021, pp. 149–154.
24. *H2O Machine Learning: Gradient Boosting Machine*. 2022. Available also from: `https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/gbm.html`.
25. WHITLEY, Darrell; KARUNANITHI, Nachimuthu. Generalization in feed forward neural networks. In: *IJCNN-91-Seattle International Joint Conference on Neural Networks*. IEEE, 1991, vol. 2, pp. 77–82.
26. KAUR, Jaspreet; KALRA, Ashima. Hybrid artificial neural network for data classification problem. In: *2017 4th International Conference on Signal Processing, Computing and Control (ISPCC)*. IEEE, 2017, pp. 66–71.

27. WONG, Tzu-Tsung; YEH, Po-Yang. Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*. 2019, vol. 32, no. 8, pp. 1586–1594.
28. POZO MONTERO, Francesc. Validation Procedures. Leave-one-out cross-validation (LOOCV) and k-fold cross-validation. 2019.
29. *Spark Machine Learning - Tuning*. 2022. Available also from: `https://rdrr.io/cran/sparklyr/man/ml-tuning.html`.
30. *H2O Machine Learning: Cross-Validation*. 2022. Available also from: `https://docs.h2o.ai/h2o/latest-stable/h2o-docs/cross-validation.html`.
31. SHEPARD, Amaya; NAHEED, Naim. Application of Data Transformation Techniques and Train-Test sit. In: *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2021, pp. 58–63.
32. MURPHY, Kevin P. *Machine learning: a probabilistic perspective*. MIT press, 2012.
33. SALLOUM, Salman; DAUTOV, Ruslan; CHEN, Xiaojun; PENG, Patrick Xiaogang; HUANG, Joshua Zhexue. Big data analytics on Apache Spark. *International Journal of Data Science and Analytics*. 2016, vol. 1, no. 3, pp. 145–164.
34. *sparklyr: R Interface to Apache Spark*. 2022. Available also from: `https://rdrr.io/cran/sparklyr/`.
35. *H2O Open-Source Machine learning framework*. 2022. Available also from: `https://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html`.
36. *Amazon Web Services Elastic Cloud Compute*. 2022. Available also from: `https://aws.amazon.com/ec2/`.

Appendix A

Processed data

Table A.1: Dataset after data processing

Variable name	Details	Type	Example
Delay	Target variable if a flight is delayed or not	String	"1" or "0"
Quarter	The scheduled quarter of the flight	Integer	1, 2, 3, 4
Month	The scheduled month of the flight	Integer	1, 2, 3, .. 12
Day of month	Day of the month in which the flight took place.	Integer	1, 2, 3, .. 31
Day of week	Day of the week in which the flight took place.	Integer	1, 2, 3, .. 7
Flight distance	Distance between the origin and destination.	Integer	650, 482, 518, 510
Seating capacity	The maximum seating capacity of the flight.	Integer	140, 196, 186, 176
Origin	Origin airport of the flight.	String	ORD, ATL, LAS, LAX
Destination	Destination airport of the flight.	String	ORD, ATL, LAS, LAX
Flight time	Total scheduled flight time of the flight.	Integer	60, 120, 135, 85
Flight speed	The scheduled average speed of the aircraft	Double	3.93, 5092.3
Carrier	Carrier number of the flight	String	AA, AS, B6, DL, EV
Planned departure local hour	Planned departure hour of the flight.	Integer	0, 1, 2, 3, .. 23
Planned arrival local hour	Planned arrival hour of the flight.	Integer	0, 1, 2, 3, .. 23
Origin wind speed	The speed of the wind at the origin.	Double	3.58, 4.56, 6.71
Origin wind direction	The direction of the wind at the origin.	Double	118.31, 91.77, 209.13
Origin air temperature	The temperature of the air at the origin.	Double	8.23, 11.25, 15.33, 10.87
Origin atmospheric pressure	The atmospheric pressure at the origin.	Double	1019.21, 1020.42, 1011.99
Origin visibility	visibility at the origin.	Double	5042.91, 871.25, 10840.76
Origin dew point	The dew point at the origin.	Double	7.54, 11.15, 13.73, 7.20
Origin precipitation	The level of precipitation at the origin.	Double	2.82, 3.36, 4.81, 2.89
Origin cloud cover	The intensity of the clouds at the origin.	Integer	100, 85, 66, 74, 12
Origin wind gust	Increase in the wind speed at the origin.	Integer	3, 12, 24, 30
Origin total snow	Total snowfall at the origin.	Double	0.0, 22.6, 7.4, 5.4
destination wind speed	The speed of the wind at the destination.	Double	3.58, 4.56, 6.71
destination wind direction	The direction of the wind at the destination.	Double	118.31, 91.77, 209.13
destination air temperature	The temperature of the air at the destination.	Double	8.23, 11.25, 15.33, 10.87
destination atmospheric pressure	The atmospheric pressure at the destination.	Double	1019.21, 1020.42, 1011.99
destination visibility	visibility at the destination.	Double	5042.91, 871.25, 10840.76
destination dew point	The dew point at the destination.	Double	7.54, 11.15, 13.73, 7.20
destination precipitation	The level of precipitation at the destination.	Double	2.82, 3.36, 4.81, 2.89
destination cloud cover	The intensity of the clouds at the destination.	Integer	100, 85, 66, 74, 12
destination wind gust	Increase in the wind speed at the destination.	Integer	3, 12, 24, 30
destination total snow	Total snowfall at the destination.	Double	0.0, 22.6, 7.4, 5.4
Origin airport congestion	Total arriving and departing flights at the origin.	Integer	100, 150, 420
Destination airport congestion	Total arriving and departing flights at the destination.	Integer	200, 250, 300