FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Predictive Analysis of Flight Times

Afonso Manuel Maia Lopes Salgado de Sousa

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ana Paula Rocha

Second Supervisor: António Castro

February 21, 2020

Predictive Analysis of Flight Times

Afonso Manuel Maia Lopes Salgado de Sousa

Mestrado Integrado em Engenharia Informática e Computação

February 21, 2020

Abstract

In airlines, flight delay is a problem that profoundly impacts the airline industry revenue every year. Flight delays are often a consequence of unexpected disruptions that consequently affects the regular operations on the airlines. Studies at several airlines have indicated that irregular services can cost as much as 3% of annual revenue and that a better recovery process could result in cost reductions of at least 20% of its irregular operations.

The impact assessment of these irregular operations is not straightforward. From detection to mitigation of a delay, several flight legs may be at risk. Some decision support systems can already attenuate the impact of delays in subsequent flights. However, there is still a substantial gap between the techniques employed in the Airline's Operational Control Center (OCC) and the ones available as industry standards in other areas. Moreover, these decision support systems work with scheduled or OCC-estimated times that are unreliable sources of information.

This thesis comprises the prediction of the aircraft movement times, namely, taxi-in, taxi-out, air time and block time. These predicted times are expected to produce more accurate data and are to be used in an OCC decision support system to replace the use of scheduled or OCC-estimated times.

The models were trained on data from TAP Air Portugal's air traffic activity from 2013 to 2018, along with scraped weather data retrieved from Iowa Environmental Mesonet. The data was analysed and refactored, along with some feature engineering. After extensive experiments, the most successful models were built, making use of a stack of linear estimators with gradient boosting as meta-estimator. This algorithm is one of the better performers in Kaggle competitions. Also, the modelling for air time and block time was split based on aircraft's fuselage size, which granted better performance.

For air time prediction, an accuracy of 90.1/64.9% within 15 minutes of deviation can be expected (respective scores for narrow-body and wide-body aircraft). Similar values were acquired for block time prediction, which outperformed the results from the currently available work in the literature addressing the prediction of block time. The prediction of taxi-in and air time seems to have never been addressed before in literature.

Resumo

Nas companhias aéreas, o atraso de voos é um problema que afeta fortemente a receita do setor da aviação todos os anos. Atrasos nos voos são frequentemente consequência de disrupções inesperadas que, consequentemente, têm impacto sobre as operações regulares nas companhias aéreas. Estudos de várias companhias aéreas indicam que operações irregulares podem custar até 3 % da receita anual e que um melhor processo de recuperação poderia resultar em reduções de custos de pelo menos 20 % das operações irregulares.

A avaliação de impacto dessas operações irregulares não é simples. Da detecção à atenuação de um atraso, vários voos podem estar em risco. Alguns sistemas de suporte à decisão já podem atenuar o impacto de atrasos nos voos subsequentes, no entanto, ainda existem lacunas substanciais entre as técnicas empregues no Centro de Controle Operacional da Companhia Aérea (OCC) e as disponíveis como padrão da indústria noutras áreas. Além disso, esses sistemas de suporte a decisões funcionam com horários programados ou estimados pelo OCC que não são fontes de informação fiáveis.

Esta tese compreende a previsão dos tempos de movimento das aeronaves, nomeadamente, "taxi-in" e "taxi-out", tempo de voo e tempo total de viagem. Espera-se que a previsão desses tempos produza dados mais precisos e são para ser usados em sistemas de suporte à decisão do OCC, para substituir o uso de horários programados ou estimados.

Os modelos foram treinados com dados da atividade de tráfego aéreo da TAP Air Portugal de 2013 a 2018, juntamente com dados meteorológicos extraídos do Iowa Environmental Mesonet. Os dados foram analisados e reformulados, juntamente com alguma "feature engineering". Após extensas experiências, os modelos mais bem-sucedidos foram construídos, utilizando uma pilha de estimadores lineares com "Gradient Boosting" como meta-estimador. Esse algoritmo é um dos algoritmos com melhores desempenhos nas competições do Kaggle. Além disso, a modelagem do tempo de voo e do tempo total de viagem foi dividida com base no tamanho da fuselagem da aeronave, o que proporcionou melhor desempenho.

Para a previsão do tempo de voo, pode-se esperar uma precisão de 90.1 / 64.9 % dentro de 15 minutos (avaliações para aeronaves NB e WB, respetivamente). Valores semelhantes foram adquiridos para a previsão do total de viagem, que superaram os resultados do trabalho atualmente disponível na literatura sobre previsão do tempo total de viagem. A previsão do "taxi-in" e tempo de voo parece nunca ter sido abordada na literatura.

Acknowledgements

I want to express my special thanks to my supervisor Ana Paula Rocha as well as co-supervisor António Castro, for their permanent availability and support throughout this work. Also, for the opportunity to collaborate with them in their startup. Secondly, I am thankful to my parents, sister, girlfriend, and closest friends for every word of encouragement that helped pushing the project forward and finalising it within the limited time frame.

Contents

List of Figures

List of Tables

LIST OF TABLES

Abbreviations

API	Application Programming Interface
ATFM	Air Traffic Flow Management
DL	Deep Learning
EDA	Exploratory Data Analysis
GIL	Python Global Interpreter Lock
GPU	Graphics Processing Unit
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
KL	Kullback–Leibler (divergence)
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MAR	Missing at Random
MCAR	Missing Completely at Random
METAR	Meteorological Aerodrome Report
ML	Machine Learning
MNAR	Missing not at Random
MSE	Mean Squared Error
NB	Narrow-body (aircraft)
NN	Neural Network
OCC	Operations Control Center
OOOI	(gate) Out, (wheels) Off, (wheels) On, (gate) In
PCA	Principal Component Analysis
RMSE	Root Mean Squared Error
SBT	Scheduled Block Times
URL	Uniform Resource Locator
VAE	Variational Autoencoder
WB	Wide-body (aircraft)

Chapter 1

Introduction

Introductory chapter to explain the motivation behind the proposal and its scope, what problem will it address and how, and a brief explanation of the document structure.

1.1 Context

In airlines, flight delay is a problem that greatly impacts the airline industry revenue every year, either by conceding millions on compensations and reschedules, or only by a loss of confidence of the passengers regarding the airlines. Several EUROCONTROL reports [EUR17, EUR18, CT15] from the past few years show an ever-increasing presence of those delays in airline's daily operations, mainly due to unaccounted or unpredictable external factors.

A survey to assess the impact of airlines' service recovery on passenger satisfaction ([ESB17]), points out that flight delay is the primary factor for passenger dissatisfaction and loss of loyalty with up to 37% of the inquired people answering that they had problems because of flight delays.

Table 1.1: Statistical reports about delay impact for 2016, 2017 and 2018 made available by the European airspace managing entity [EUR17, EUR18].

	2016	2017	2018
Arrival Punctuality (%)	81	79.7	75.8
Average Delay per Flight - Reactionary (min)	5.2	5.5	6.7
Departure Delay (min)	11.1	12.4	14.7

* The values in the table are average annual values.

As shown in table 1.1, flight delay is a relevant problem in airlines and does not seem to be slowing down.

Despite the increasing capabilities of today's hardware and promising results in decision support systems, there is still a considerable gap between the available technology and the techniques employed in the airlines' operations control [CLLR10, dC13]. A large number of airlines have determined that irregular operations can have a cost of up to 3% of their annual revenue and that a better recovery process could lead to a cost reduction of at least 20% of the current cost with disruptions [as cited in dC13].

According to IATA [IAT18], the global net profit of commercial airlines was of 32.3 billion dollars, meaning almost 200 million dollars could be saved annually if a proper recovery process was applied worldwide. Moreover, a study conducted by Westminster University for EUROCONTROL [CT15] attributes a cost of 81€ per minute of delay. For a departure delay value of 14.7 minutes (table 1.1), flights in 2018 had an average delay cost of almost 1200€.

1.2 Motivation and Objectives

Before the departure of an aircraft, a sequential approach for operational plan-making is applied complying to numerous restrictions and rules: rules on aircraft maintenance, regulations on maximum flying times for the crew members, airline-specific regulation, among others. According to [CLLR10], this plan is given to the Operational Control Center (OCC) a few days before the day of the operation.

At TAP, the operational flight plan can be accessible up to 6 hours before the scheduled departure time of two aisle planes (wide-body [WB]), and up to 2 hours before the scheduled departure time of one aisle planes (narrow-body [NB]). As the departure approaches, new versions of the operational flight plan are issued.

Because the information on the flight plan is compliant with all restrictions and rules required by the entities that control the airspace, this is deemed as the most accurate information and, when available, may be used by a decision support system and useful suggestions are expected.

The problem that arises is that any decision support systems introduced in the OCC will be fed with inaccurate information if the disruption assessment is made just a few hours earlier than the flight's departure because of lack of proper information - flight plan's information.

This problem motivates the prediction of the partitioned aircraft movement.

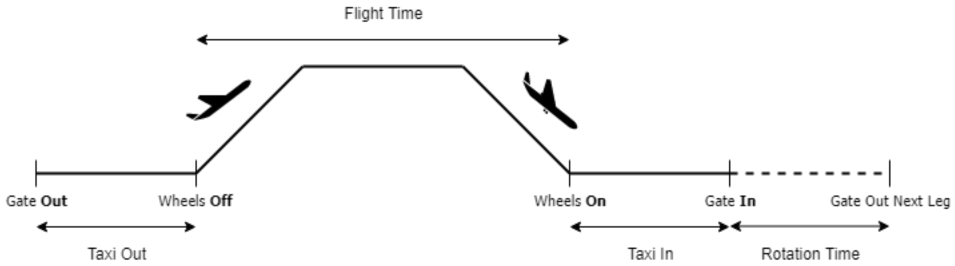


Figure 1.1: Partitioned aircraft movement (simplified).

In figure 1.1, a comprehensive diagram of the actual aircraft movement times is illustrated. The accuracy of these times is central for a correct disruption impact assessment. Meaning, to overcome the lack of proper information to feed to decision support systems in the OCC, models capable of predicting the flight movements may be able to help decision support systems and, consequently, OCC operators to assess in advance, with a high degree of certainty, the impact of a disruption.

The object of this work is to develop four predictive models whose targets are *taxi-in*, *taxi-out*, *air time* and *actual block time*. Depending on the available information on the flight plan, one or more predictive models could be used by the decision support systems. The task assumes a period between delay detection and flight departure of up to 24 hours.

1.3 Dissertation Structure

This document is organised as follows. The current chapter, **Introduction**, outlines the understanding of the topic and its importance in the aviation scope. Chapter 2, **Bibliography Review** analyses similar works and presents a brief introduction to the general concepts underlying data mining problems. Chapter 3, **Problem Description**, describes the addressed problem and analyses the initial dataset, discussing its content and cleanup procedures, as well as encompassing the data analysis. Chapter 4, **Feature Engineering**, presents the data refactoring to expose the relationships between data for model performance improvement explicitly. Chapter 5, **Modelling**, shows algorithm parameter tuning and result scrutiny. Finally, Chapter 6, **Conclusions and Future Work** summarises the developed work and enumerates future directions of work.

Chapter 2

Bibliographical Review

This chapter shows the concepts and techniques employed in similar problems, along with an overview of the commonly used methodologies for these kinds of tasks. This chapter serves as a foundation to base assumptions for the work described in the following sections.

2.1 Background

This work will be developed by borrowing techniques and approaches present in the literature on data mining tasks.

Data Mining Data mining is a subfield of computer science, statistics and artificial intelligence, which goal is to extract information of complex structures in the data (with intelligent methods), transform it and serve it for further use [HTF01].

2.1.1 Supervised Learning

In Data Mining, the primary learning mechanisms are supervised learning, unsupervised learning, and reinforcement learning. The work reported in this dissertation can be classified as a supervised learning task.

Supervised learning is a predictive mechanism that consists in learning a function that, using input-output pairs, maps inputs to outputs. It is named after the requirement for the data to be clearly labelled, and thus the result of the output can be supervised and assessed whether it is correct or incorrect.

Essential to the process of supervised learning is the concept of training (iterative improvement in learning). In this training task, the data to feed the model with (dataset) is split into training and testing sets, and possibly a validation set. A learning algorithm is fitted in the training data to learn the core aspects of the data while keeping the ability to generalise to unseen data. The validation set is used alongside the training set as a method to validate the effectiveness of the fit to that training set, which result would be used to tune the learning parameters of the algorithms (hyperparameters). The test set serves as a previously unobserved set of data to assess the generalisation capabilities of the model. After training a model, a new data item can be inputted into that model to return an inferred output.

Regression vs Classification

Regression and classification are the two primary learning tasks in supervised learning problems. Fundamentally, classification is about predicting a discrete class label and regression is about predicting a continuous value.

In classification, the output variables are often called labels or categories and the mapping function that infers outputs given inputs predicts the class or category for a given observation. A typical example of classification comes with detecting spam emails, classifying whether an email is spam or not.

In regression, the output of a learning task is a real-value where the mapping function predicts an integer or floating-point value for an observation.

These problems can be extended to include the prediction of more than one output. This work will only focus on single-output regression problems.

2.1.2 Assessing Regression Quality

Regression Evaluation Metrics

In a regression problem, a model's performance assessment is generally made using **residuals** - difference between observed and predicted values.

Let y_i be the truth value and $\hat{y}_i$ the model output (or predicted value), for a sample i. The residual would be:

$$e_i = y_i - \hat{y}_i$$

A naive approach to assess a model's performance would be to compute the mean error $\sum_i \varepsilon_i$, but it turns out that positive and negative errors cancel each other out, giving no true knowledge of how well the model performs. Therefore, more sophisticated metrics were developed. The following are the commonly used in regression problems:

- **Mean Absolute Error (MAE):**

$$\frac{1}{n}\sum_{i=1}^{n} |e_i| \tag{2.1}$$

- **Mean Squared Error (MSE):**

$$\frac{1}{n}\sum_{i=1}^{n} (e_i)^2 \tag{2.2}$$

- **Root Mean Squared Error (RMSE):**

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(e_i)^2} \tag{2.3}$$

- **Mean Absolute Percentage Error (MAPE):**

$$\frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{e_i}{y_i}\right| \tag{2.4}$$

- **Coefficient of determination (R^2):**

$$1-\frac{\sum_{i=1}^{n}(e_i^2)}{\sum_{i=1}^{n}(y_i-\bar{y})^2} \tag{2.5}$$

MAE, MSE and RMSE express average prediction error in variable units. MAPE expresses average prediction error in percentage, normalised by observation, meaning the outcome value is based upon true observation. These metrics can range from 0 to ∞ (even MAPE, which is expressed as a percentage), are indifferent to the direction of errors and negatively-oriented, which means lower values are better than greater ones. R^2 expresses the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1 (0 to 100%), and it is positively-oriented (higher values are better).

MAE measures the error magnitude in a set of predictions, assuming equal weights. Being in the same target units of the real values gives the metric a very intuitive meaning. MSE and RMSE square errors before they are averaged, which gives higher weights to more significant errors. This property makes these metrics more useful when significant errors are particularly undesirable. In general, RMSE is more popular than MSE because it is more interpretable in the target units.

MAPE puts a more substantial penalty on negative errors than on positive ones and is problematic if the target variable has zero-valued observations. Nevertheless, MAPE is helpful for positive target value sets because of its very intuitive interpretation in terms of relative error. R^2 is interesting because the value denotes the variability of the dependent variable that was accounted for, although a residual plot should support its analysis because high R^2 values can be misleading.

Residual plots

These plots show the predicted value on the $x-axis$, and the residuals on the $y-axis$. When looking at these plots, ideal solutions would have their points scattered in a "random" cloud pattern - preferably clustered close to the zero lines (i.e. smaller residuals). Non-ideal solutions would show patterns where the residuals either change in proportion with the predicted values (i.e. higher absolute value, higher residual) or any non-linear patterns, commonly, either a U or upside-down U shapes. Also, when points are not evenly scattered, whether along the $y-axis$ (i.e. outliers) or along the $x-axis$, can be a sign of existing outliers or clusters.

2.1.3 Data Preprocessing

Some popular techniques are often used to prepare the data for the modelling process. Almost all of them address the over-fitting problem or are required for the proper implementation of certain algorithms. The following sections describe some of these techniques.

Feature scaling

Feature scaling is a technique to standardise the importance of predictors across datasets of varying scales of features. A popular approach is normalisation or **MinMax scaling**, where the data is scaled to a fixed range - usually 0 to 1 - which can lead to small standard deviations and consequently, the suppression of outliers. An alternative to normalisation that is generally preferable is Z-score normalisation (or standardisation) or **Standard scaling**, which makes the dataset have a mean of 0 and a standard deviation of 1. A more robust scaler to non-normally distributed predictors is **Robust scaling**, in which, rather than scaling to unit variance from mean value (which can be negatively influenced by outliers or distribution skewness), the scaling is made in the interquartile range from the median value.

Categorical encoding

Some machine learning algorithms can learn directly from categorical data with no data transformation required, but many cannot operate on categorical data directly. They require all input and output variables to be numeric.

If the target variable is categorical, it may be required to convert model predictions back into nominal form in order to present them, use them in some application or even assess the predictive capabilities of model training.

Label Encoding Label encoding assigns, to each category, a value from 0 through $n - 1$, where n is the number of distinct labels for each property. This type of encoding makes models misunderstand the data to be in some kind of order and infer relationships between variables incorrectly.

One-Hot encoding The drawback of Label encoding can be overcome using one-hot encoding, which maps each category to a binary vector with its truth value set by the presence or absence of the feature. Meaning, it splits each categorical variable column into multiple binary columns, one for each unique label. For each new column, the values are replaced by 1s or 0s whether the observations have the column-represented value or not. This method may produce many columns if the number of unique labels is very high for each feature and consequently slow down the learning process significantly.

Target Encoding In target encoding, for each category, the new label is dependent on the target's mean value. Similarly to label encoding, target encoding does not increase data's volume, which is particularly helpful for faster learning. Because this encoding method brings out the relation between similar categories, it can lead to over-fitting, and the use of this encoder is advised to be under cross-validation.

Leave One Out Encoding It uses a similar approach to target encoding but is more robust to outliers, achieving this by excluding the current row's target when calculating the mean target. This algorithm is particularly useful for high cardinality categorical features.

Cross-Validation

Cross-validation (CV) is one of the most widely used methods for estimating the predictive capabilities of a learning model in new data. It uses part of the available data to fit the model, and the rest to testing. It consists in splitting the training set into K number of similar-sized subsets, called folds, and iteratively fitting the model, training it on $K-1$ folds of data and evaluating on the k-th fold (called the validation data) [HTF01].

Cross-Validation helps mitigate the over-fitting problem and more accurately assess the generalisation capabilities of a model. The final performance metrics are computed using the average of the performance on each fold.

2.1.4 Regression Algorithms

Linear algorithms

In statistics, linear regression is an approach to modelling the relationship between a scalar value (dependent variable) and one or more predictors (independent variables). For an input vector $X^T = (X_1, X_2, \dots, X_p)$, the linear regression model has the form:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{2.6}$$

For a real-valued output Y, the regression model assumes $E(Y|X)$ is linear or a reasonable approximation.

Some algorithms try to improve on the linear regression model like Ridge, Lasso or Elastic Net.

Ridge Ridge Regression is a method of regularisation of a linear least squares estimator. It shrinks the regression coefficients by imposing a penalty on their size [HTF01]. This penalty can alleviate the problem of multicollinearity in linear regression, commonly present in models trained on large numbers of features. It does so by adding the squared magnitude of the coefficients as penalty term to the loss function (also known as L2 regularisation, or weight decay on neural networks). This penalty introduces a tolerable amount of bias that significantly decreases variance. In general, this method provides improved efficiency in parameter estimation problems by avoiding the over-fitting issue.

Lasso Works similarly to Ridge Regression but instead of applying L2 regularisation, applies L1 regularisation that consists on adding the absolute value of the magnitude of the coefficients as penalty term to the loss function, resulting in a shrinkage of the less important feature's coefficient to zero and thus, removing some features altogether. Lasso works well for feature selection in case of problems with vast numbers of features.

Elastic Net Elastic Net is a regularised regression method that linearly combines the L1 and L2 penalties of the Lasso and Ridge methods. Elastic Net selects variables like Lasso and shrinks the coefficients of correlated predictors like Ridge [HTF01].

Ensemble learning algorithms

Ensemble methods use multiple estimators to obtain better predictive performance than any of those estimators alone. An ensemble of models is a set of models that combined their predictive capabilities in some way to improve the evaluation of new examples [Die00].

Random Forest Random forests are a combination of tree predictors formed by sampling from the training data, training the same tree many times and averaging the result [Bre01, HTF01]. Trees can capture complex structures in data but are notoriously noisy so benefit significantly from averaging, which reduces the variance. Also, if big enough, trees can have a relatively low bias.

Gradient Boosting Gradient Boosting constructs a prediction model in the form of an ensemble of weak prediction models (decision trees). The goal is to minimise a cost function that is constrained to be a sum of trees [HTF01]. It randomly samples trees where each is constructed and trained to reduce the residuals generated by the previous one [DYC+20].

XGBoost XGBoost follows the principle of gradient boosting but uses a more regularised model formalisation to control over-fitting, which generally gives it better performance. Is one of the algorithms often present in winning solutions at the machine learning site, Kaggle [1] [CG16].

Stacking Stacking is an ensemble technique to combine multiple regression models to give improved prediction accuracy [Bre96]. The model is composed of two levels, the first level of weak learners that are fit in a cross-validation fashion, and a second level estimator that used the resulting predictions of the first level of estimators as input data for improved accuracy.

[1] https://www.kaggle.com/

2.1.5 Neural Networks

Other algorithms that can be adapted for regression problems are the now popular Neural Networks (NN). These are the backbone of Deep Learning and consist of multi-layer networks of neurons that are used to make predictions.

The resurgence of NN in these past few years is due to three factors:

- Big Data - the data is now much more easily obtained and in much larger quantities. Also, storage has become more accessible and cheap, as well as the development of data indexing tools that aid in this task.
- Hardware - neural networks are massively parallelizable [HZL+13] and can benefit tremendously from modern Graphics Processing Units (GPU) architectures.
- Software - due to newly risen powerful and open-source tools, building and deploying DL algorithms is streamlined and accessible for everyone.

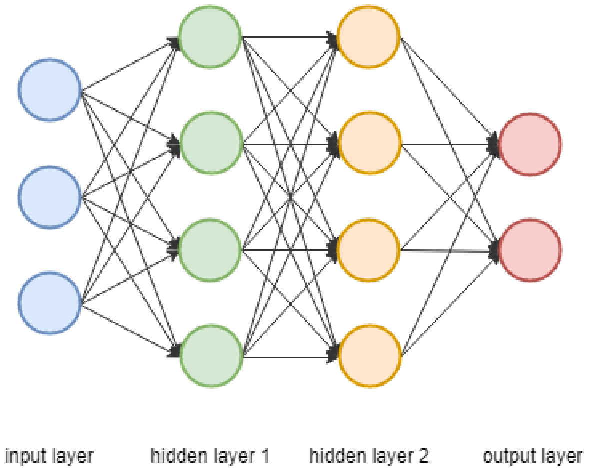


Figure 2.1: Neural network with two hidden layers

Figure 2.1 shows the structure of a neural network with two hidden layers. The nodes from one layer are fully connected with the nodes of the next layer. The network learns by propagating data from the input layer to the output layer, and back again to learn from its "mistakes".

Forward Pass Figure 2.2 illustrates the forward pass process. This behaviour works for each layer of the network. The next neuron's activation is computed as the product sum of the node values with the edges' weights. The obtained value is passed through an activation function that introduces non-linearities into the network, meaning, that activation function allows the network to approximate to arbitrarily complex functions.

Backward Pass The backward pass is the reverse of the forward pass. However, instead of calculating and propagating the node activations to the next layer, it calculates the errors. It tweaks

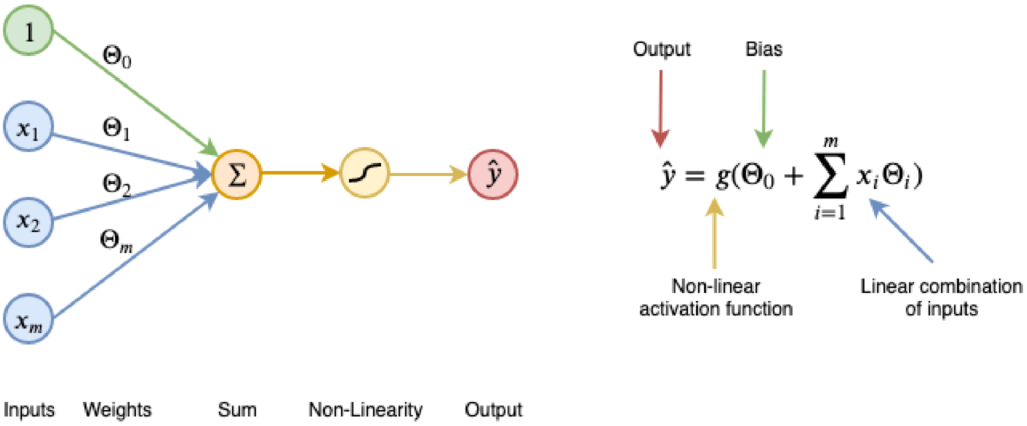


Figure 2.2: Single output neuron forward propagation

the weights in the direction of error minimisation (gradient descent), from the output layer up until the input layer. These tweaks are done computing the gradient of the cost function (e.g. Root Mean Square Error), which tells how the cost function changes for each weight.

Batching Typically, the gradient is computed using mini-batches which are subsets of the whole dataset. Computing the average gradient across the data points from a mini-batch and using that as an estimate of the true gradient is much more inexpensive than using the whole dataset at once.

Regularization Regularisation techniques are meant for constraining a model in such a way that it is discouraged from becoming too complex and just memorising the training data, also known as overfitting. Some popular approaches are Dropout and Early Stopping.

2.2 Related Work

Predictive approaches for delay minimisation in airlines are an important ongoing research area due to plane travel being increasingly sought after. Also, the unpredictable nature of delays that are caused due to unanticipated circumstances motivated researchers to address this problem using data analytics.

2.2.1 Aircraft movement data prediction

By 2005, the propagated consequences of late-arriving flights on the following segments [as cited in Coy06] were already noticed.

This motivated the work of Steve Coy that, in 2006, developed a statistical approach using linear regression to predict block times [Coy06]. The experience was made using 2 million US domestic flights by six airlines in 2004 and comprises the construction of a model for each airline. This work achieved a high level of significance with a R^2 of over 95% for every built model. Models were built for each airline using information about frozen precipitation and the estimated impact of various meteorological events on the air traffic. Coy concluded that these were great block time predictors.

In 2010, Balakrishna *et al.* developed a reinforcement learning approach for predicting aircraft taxi-out times [BGS10]. The problem was modelled building an objective function that minimises the absolute value of taxi-out residuals. The predictions were made 15 minutes before departure. The rationale for setting this threshold was that the prediction of flights too much in advance of gate pushback time (i.e. off-gate time) is likely to be less accurate. This research got great results in predicting the mean taxi-out time as well as the taxi-out for specific flights, with respectively, 94% accuracy within 1.5 min deviation and 81% accuracy within 2 min deviation. This work only used OOOI times ((gate) Out, (wheels) Off, (wheels) On and (gate) In).

These two works mentioned above predict with a high degree of certainty two of the four targets of this work. As is a supervised learning approach, it should be more reliant on the "goodness" of data.

2.2.2 Delay prediction

The prevalent flight delay in airline's operations led the research field to focus mainly towards developing approaches that address flight delay prediction. Several studies have been conducted to predict aircraft delays, from departure and arrival.

In 2014, Rebollo and Balakrishnan developed models using a Random Forest algorithm to predict the existence of departure delays and their values, 2 to 24 h advance [RB14]. The regression approach of this work got an average median test error from 20 min to 27.4min, depending on the forecast horizon.

in 2016, Choi *et al.* [CKBM16] implemented a Random Forest for predicting existence of arrival delay. This implementation got an accuracy of 80.4%. The authors proposed for future work a cost-sensitive approach because of a decline in the accuracy when sampling was applied. The models hardly predict the minor class. The author also noted that some algorithms were not able to make predictions when trained without weather data. In the next year, following this work, the same authors develop the cost-sensitive approach they previously proposed [CKBM17], using the same algorithms, data and techniques. The idea is to weigh more severely misclassifications of on-time as delay, and more forgivingly misclassifications of delay as on-time. However, applying this technique degraded the models' performance.

In 2017, Manna *et al.* developed an approach for predicting arrival and departure delays in flights using Gradient Boosting [MBK$^+$17]. In this work, the model was built using historical flight data from 70 United States airports. That data was retrieved from the US Department of Transport. This model showed good results with coefficients of determination for arrival and departure delays over 92%. Interesting to note a 94% Pearson correlation coefficient between arrival and departure delays, showing no mitigation in the delay propagation.

In the same year, Thiagarajan *et al.* [TSS$^+$17] used multiple machine learning algorithms and a Deep Neural Network to predict the occurrence of a delay in a specific flight (through a classification problem) and to predict its numerical value (through a regression problem). Gradient Boosting Classifier and Extra Trees Regressor performed the best from the set of supervised learning algorithms.

2.2.3 Deep learning approaches

With the tremendous increase on the size of the available data in all kinds of industry segments and latest studies showing neural networks and deep architectures yielding outstanding results in various data mining problems, these techniques became popular and are now being tested in the aviation field as well.

In 2015, Hugo Alonso and António Loureiro used a 5-layer NN to predict departure delays in Porto airport [AL15]. Chakrabarty et al. [CKD$^+$19]. Variable importance is referenced in this work, and the solution with the NN only required two variables to explain +99% of the variance in data.

In 2016, Khanmohammadi *et al.* [KTK16] had used multi-level artificial neural networks for predicting arrival delay time in JFK airport. This study was conducted with a small dataset and only with flight information. It achieved an RMSE of 13.7%.

In the same year, Kim *et al.* [KCBM16] applied deep LSTM RNN architecture to build model for departure delay prediction. The developed model achieved an accuracy of 87.42% in predicting individual flight delays. The employed architecture used a 5-layer NN with stacks of LSTM cells modelling data as 9-day sequences of day-to-day activity. Weather conditions at origin and destination airports were presumed to be important factors for the prediction task.

In 2019, Yu *et al.* [YGA$^+$19] made another attempt on Deep Learning usage for flight delay prediction. A DBN–SVR prediction model was employed, where the Deep Belief Nets extract the hidden factors that impact flight delays while reducing dimensionality and eliminating redundant information. The output of DBN is then used as the input of the Support Vector Machine (SVM) to capture the key influential factors and generate the delay predictions. 99.3% of the predicted values were within 25 min deviation from the actual ones.

2.3 Key Findings

In the existing literature, researchers worked extensively on flight delay problems of classification and regression, but not so much on the prediction of any of the aircraft movement times.

Table 2.1 shows an aggregated visualisation of the explored works. Information on developed classifiers was trimming when their regressor counterpart was also developed.

Weather conditions are significant air traffic data predictors. In Steve Coy's statistical approach to predicting block times [Coy06], it was concluded that weather conditions are good block-time predictors. Later, in Choi *et al.* work on weather-induced airline delay [CKBM16], it was stated that some algorithms had no predictive capabilities when trained with datasets without information on weather data. Further, several literature approaches seem to have used some kind of weather data to achieve good results.

Table 2.1: Overview of literature studies and main remarks.

Authors	Year	Targets	Best Algorithm	Dataset	Evaluation	Remarks
Steve Coy	2006	block time	Linear Regression	historical flight and weather data	$R^2 = 95\%$	linear models can be strong learners; weather conditions are significant block time predictors
Balakrishna *et al.*	2010	taxi-out	Reinforcement Learning	OOOI data	acc = 93.7%	just OOOI data can train a strong learner
Rebollo & Balakrishnan	2014	delay at departure	Random Forest	historical flight data	avg. median = [20, 27.4]min	deterioration of performance is dependent on forecast horizon; ensemble techniques had superior performance over others
Hugo Alonso & António Loureiro	2015	delay at departure	NN	historical flight and weather data	-	the variable importance is really high for a couple of predictors
Khanmohammadi *et al.*	2016	delay at arrival	Multi-level NN	historical flight data	RMSE = 0.137	multi-level NN can minimise multicollinearity of one-hot encoding
Choi *et al.*	2016	delay at arrival	Random Forest	historical flight and weather data	acc = 80.4%	weather conditions are significant delay predictors; ensemble techniques had superior performance over others
Kim *et al.*	2016	delay at arrival	LSTM Recurrent Neural Network	historical flight and weather data	acc = 86%	successful approach with a deep architecture
Choi *et al.*	2017	delay at arrival	Random Forest	historical flight and weather data	AUC = 64%	cost-sensitive approach for weighted classification not good for unbalanced data
Thiagarajan *et al.*	2017	delay at departure and arrival	Extra Trees	historical flight and weather data	$R^2 = 94.4/98.5\%$	ensemble techniques had superior performance over others
Manna *et al.*	2017	delay at departure and arrival	Gradient Boosting	historical flight data	$R^2 = 94.8/92.3\%$	boosted techniques deliver good performances
Chakrabarty *et al.*	2018	delay at arrival	Gradient Boosting	historical flight data	acc = 79.7%	boosted techniques deliver good performances
Yu *et al.*	2019	delay at arrival	DBN-SVR	historical flight and weather data	$R^2 = 93\%$	air route situation can be used instead of weather conditions

Ensemble learning approaches seem to be the best for air traffic data. In Rebollo and Balakrishnan work, it was said that "Random Forests were chosen due to their superior performance" [as cited in RB14]. In both Choi *et al.* works [CKBM16, CKBM17], from various algorithms, Random Forest was the best one. In Manna *et al.* work [MBK+17], the Random Forest was not used because it only reduces variance, and a Gradient Boosting was used instead because it addresses bias and variance simultaneously. Thiagarajan *et al.* also used a Gradient Boosting classifier [TSS+17] to achieve great results. In a regression task in this same work, an Extra Trees regressor got the best performance. Lately, it seems that ensemble with boosting is getting popular in this field of research as is for other research segments.

Chapter 3

Problem Description

This section contains information about the problem to be addressed, the prediction of flight times, showing its relevance and how does it fit in the scope of airline operations. It also discusses the dataset used, comprising historical flight and weather data. Then, it describes the data analysis and the cleanup procedure performed in this dataset.

3.1 Problem

3.1.1 MASDIMA

Reactionary delay is the major type of delay in airlines and is caused by the late arrival of an aircraft or crew from previous flights. This delay may spread to several subsequent flights. The event that triggers the delay can be identified and its propagated effects assessed. One software that does this is MASDIMA.

MASDIMA (Multi-Agent System for Disruption Management) is a multi-agent system that monitors unexpected events, assesses the impact of disruptions and provides a timely solution to solve them.

The software assigns different agents with optimisation tasks to mitigate the impact of the detected event. Each agent solves for a dimension of the airline's environment - aircraft, crew or passenger [CRO14]. The agents deliberate on their own on how to optimise the given task and finally cooperate to come up with a timely solution. This integrated solution is achieved through Q-Negotiation [dC13, CRO14].

This system is intended to be used by the Operations Control Center (OCC) to help them in their decision-making.

The problem that arises is that MASDIMA, typically, only has the scheduled times to work with when a delay is detected. More accurate predictions, from the operational flight plan, for example, are available only a few hours before the flight schedule time of departure.

3.1.2 Scheduled time problem

Typically, airline set scheduled block times (SBT), for a certain flight, as much as six months in advance [Coy06, DA12]. Accounting for the always-prevailing uncontrollable events in the airlines' operations, these render the scheduled times' usage unreliable.

Moreover, the setting of SBT is prominently set to achieve marketing advantages. A shorter SBT enables more competitive scheduling and lowering of resource expenditure - airline resources are always being used, which enables a more significant number of flights to be made in the same period. A longer SBT means better on-time performance (keep up to the schedule) and a lesser propagated delay.

According to Hao [HH14], airlines are willing to experience occasional severe delays in exchange for a shorter SBT, although this may lead to other inconveniences. In a study conducted in 2010 sponsored by the American Federal Aviation Administration [BBD^{+}10] it was noted that, since airline fleet and crew schedules are mainly based on the scheduled times, excessive or even moderate amounts of flight delays can be highly disruptive causing extra crew costs, various costs associated with accommodating disrupted passengers and even aircraft repositioning.

On the other hand, airlines, generally large ones, can lengthen their SBTs as padding for heavy traffic. Further, airlines create their fleet plans based on the scheduled flight arrival and departure times so that increasing the scheduled buffer leads to changes in schedules and eventually to poor aircraft utilisation [HH14, BBD^{+}10].

This misleading schedule generates annual misleading statistics that do not adequately translate the efficiency of the service. As airlines try to seize a competitive edge over the rest of the market, the SBTs are unreliable to provide information about the actual aircraft movements, which renders the scheduled times useless as a reliable source of information. As the departure date comes near, this problem is, to some degree, mitigated - using the flight plan.

Depending on how early the document is made available, key information might be missing. When this data is nonexistent, estimated times are used instead. Generally, they are not computer-generated, but built by the OCC operators based on experience and *rules of thumb* [dC13]. The most critical information to assess flight delays is the various partitions of aircraft movement.

3.1.3 Flight Time Decomposition

Along with the SBT, the partitioned times in figure 1.1 take part in the full flight time decomposition. These are all the segments of travelling that an aircraft may perform in its activity for an airline. Figure 3.1 illustrates all the relevant times in a flight.

Taxi-out is the time between an aircraft exit from the gate until it takes off. Analogously, the taxi-in is the time from landing to park. The time between scheduled and actual departure time is defined as departure delay or gate delay. Arrival times work analogously. Effective flight time is the total time of resource availability.

SBT is the time duration between the scheduled departure and scheduled arrival times. The actual block time (flight time) is the time between actual departure and arrival time.

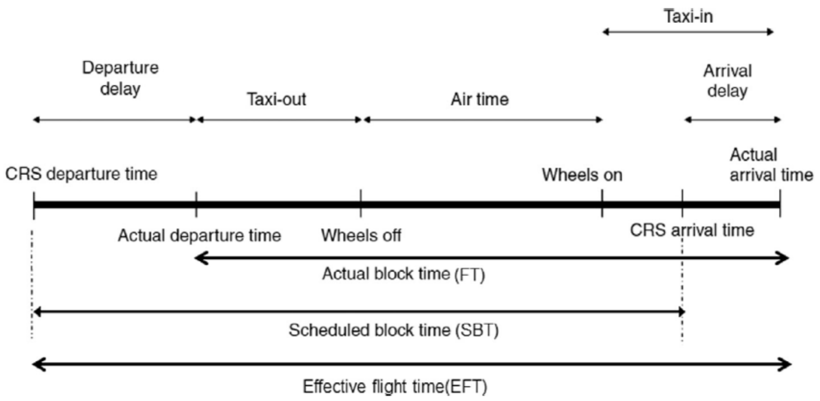


Figure 3.1: Full flight times decomposition [DA12].

Another important time is the rotation time or ground time, which is the time between actual arrival and the next leg's departure time.

3.2 Data Source

For this work, a dataset with historical flight data was provided by TAP Air Portugal. This dataset did not include weather data which had to be extracted from a different source. Following is the raw data pre-processing and compilation made to form a first stage dataset, encompassing web data extraction and data refactoring.

3.2.1 Flight Historical Data

The data available was provided by TAP Air Portugal [1]. This dataset is made up of historical flight data of TAP aircrafts' activities from 2013 to 2018. The raw data consists of 822264 records with 35 attributes.

It was prior knowledge (from the flight data provider) that some observations of the data were incorrect. Records were containing older data from already existing flight records, meaning, duplicate records with outdated information and also instances reporting land vehicles' activities. Both the former and the latter were removed.

After preliminary refactor, the dataset ended up containing 591152 records and 35 attributes. The features of this dataset are contemplated in table 3.1.

Trying to minimise the strain of iterating over almost 600000 records and 35 attributes, the redundant information that stood out was removed. From the attributes that were initially made available, the following were considered irrelevant:

- carriage code: It was the same for all observations.
- taxi times: Can be extracted from actual times and be more accurate.

[1] https://www.flytap.com/

Table 3.1: Flight historical data features by data type. Data granted by TAP Portugal.

name	data type
Flight Date	date
Carriage Code	string
Flight Number	int
Tail Number	string
Aircraft Type Code	int
Aircraft Description	string
Aircraft Family Code	string
Departure Airport	string
Estimated Flight Time	time
Taxi out	time
Taxi in	time
Scheduled Time of Departure	datetime
Estimated Off-Block Time	datetime
Actual Off-Block Time	datetime
Estimated Take-Off Time	datetime
Actual Take-Off Time	datetime
Arrival Airport	string
Estimated Landing Time	datetime
Actual Landing Time	datetime
Estimated On-Block Time	datetime
Actual On-Block Time	datetime
Scheduled Time of Arrival	datetime
Business Seats	int
Economy Seats	int
Business Sale Seats	int
Economy Sale Seats	int
Flight Leg Status	char
Service Type	char
Aircraft Owner Code	string
Time of Delay Registration	datetime
Minutes of Delay	int
Delay Remarks	string
Delay Code	int
Scheduled Time of Arrival	char
Delay Description	string

- estimated date times: There is no background knowledge about how were the estimated times calculated - may introduce bias in the data.

- seat info: No important inferences can be made using the flight occupancy because the *economy seats* feature is zero-filled and the rest serve no purpose as standalone features.

- leg status: It was the same for all observations.

- delay descriptions: The *delay remarks* and *description* were inconsistent and redundant since the *delay code* contains their information.

The number of attributes are now: **35 -> 20**.

3.2.2 Weather Data Extraction

As concluded in section 2.3, the weather has a great impact on aircraft operations and weather data is a major predictor of the various flight times. Therefore, weather data must be obtained to complement the already existing flight historical data.

The weather information was retrieved from Iowa State University's website [Uni]. This university maintains an archive of airport weather observations from around the world which is extracted using a mesonet - a network of automated monitoring stations built to observe weather and environmental phenomena.

Weather Attributes

The website's available data contains several attributes from which the ones deemed as most important for the airline domain were selected. Table 3.2 briefly describes each one.

Table 3.2: Web-retrieved weather features [Uni].

name	units	description
air temperature	Celsius	Air Temperature in Celsius, typically at 2 meters
wind direction	degrees	Wind Direction in degrees from north
wind speed	knots	Wind Speed in knots
visibility	miles	Visibility in miles
wind gust	knots	Wind Gust in knots
cloud coverage level 1	-	Sky Level 1 Coverage
cloud coverage level 2	-	Sky Level 2 Coverage
cloud coverage level 3	-	Sky Level 3 Coverage
cloud height level 1	feet	Sky Level 1 Altitude in feet
cloud height level 2	feet	Sky Level 2 Altitude in feet
cloud height level 3	feet	Sky Level 3 Altitude in feet
ice accretion 1h	inches	Ice Accretion over 1 hour in inches
ice accretion 3h	inches	Ice Accretion over 3 hours in inches
ice accretion 6h	inches	Ice Accretion over 6 hours in inches
peak wind gust	knots	Peak Wind Gust in knots
peak wind direction	degrees	Peak Wind Gust Direction in degrees from north

The attributes from table 3.2 where selected with the help of a domain expert that narrowed down the list of attributes to the fundamental ones for the task.

The available data on the website was extracted from METAR (Meteorological Aerodrome Report) data, which is a format for reporting weather information. This format is compliant to some rules that help understand and define boundaries for the data. The reporting standards for the METAR format on the retrieved features are as follows:

- The wind direction and wind speed values are calculated as means from the 10 minutes immediately preceding the observation.
- Wind gust is registered if wind speed exceeds 10 knots and is registered as the maximum wind speed for the 10-minute window. Peak gust and direction work analogously but for 24 hours.
- The visibility has a lower bound of 50, meaning 50 or fewer meters of visibility and an upper bound of 10000, meaning visibility of 10 kilometres or more (noting this value is in miles).
- The cloud levels correspond to the prominent layers of clouds being the level 1 the lowest and level 3 the highest.

Cloud coverage is reported in oktas and estimates how many eighths of the sky are covered in clouds. Intuitively, this unit ranges from 0 oktas (clear sky) through to 8 oktas (completely overcast). Complete coverage reporting abbreviations are in table 3.3.

Table 3.3: Cloud coverage reporting abbreviations.

abbreviation	meaning
SKC	No cloud/Sky clear
NCD	Nil Cloud detected - automated METAR station has not detected any cloud, either due to a lack of it, or due to an error in the sensors
CLR	No clouds below 12,000 ft (3,700 m)
NSC	No (nil) significant cloud - none below 5,000 ft (1,500 m)
FEW	Few - 1–2 oktas
SCT	Scattered - 3–4 oktas
BKN	Broken - 5–7 oktas
OVC	Overcast - 8 oktas, i.e., full cloud coverage
VV	Clouds cannot be seen because of fog or heavy precipitation, so vertical visibility is given instead

Web Scraper

The data was available in tabular format and accessible through query strings from the university's website [Uni]. The information was collected automatically using a web scraper. This web scraper was developed in Python, making use of already existing packages, namely, *urllib*. This package is a standard python library that provides several modules for working with Uniform Resource Locator (URLs). Those modules define functions and classes which help in opening URLs and enable easy extraction of content.

The task of retrieving weather data was performed twice for each observation in the flight data — one time for the origin airport's weather and another for the destination airport's weather.

The website's API (Application Programming Interface) minimum extraction size is a day's worth of observations which required trimming on the extracted data. The only weather records kept were the ones with the closest registered time to the respective threshold. For the origin airport's weather, the record kept was the one with the closest registered time to the off-block date of the observation. For the destination airport's weather, the record kept was the one with the closest registered time to the on-block date of the observation.

The data was clean and tabular formatted from the source, so little to no parsing was required. Both the origin and destination weather information retrieved for each observation of the flight on-time performance dataset was concatenated with the respective record.

The number of attributes is now: **20 -> 52**.

Optimisation and Parallel Programming

The process described in the previous section was computationally intensive and time-consuming. Each iteration required two API calls and data processing. The download speed is also a huge setback when making over 1 million calls.

Parallel programming was used to try to accelerate the task. In Python, parallel programming is limited because of the existence of GIL. GIL or Python Global Interpreter Lock is a mutex that allows only one thread to hold the control of the Python interpreter. Meaning, multithreading was not an option and multi-processing was used instead (in a quad-core machine). The task was divided in chunks of 5000 records that were individually executed, parsed and locally stored. After the execution of every chunk, the stored files were concatenated.

3.3 Data Visualisation and Cleanup

Data visualisation gives insight about the data: discover patterns, spot anomalies or test assumptions. Ultimately, the purpose is to expose the underlying features of the data and acquire knowledge to serve as a foundation for the modelling task.

Joining Exploratory Data Analysis (EDA) with data cleaning presents change iteratively and can more closely represent the programmatic approach used.

In some instances where there are many data points to plot, a sample of September is used because it has characteristics that represent the annual average behaviour of operations, as noted in [dC13].

Table 3.4 shows the first step on data analysis with a per-attribute overview, that identifies the name of the attributes, their type, example values, number of existing distinct values and the missing ratio.

Table 3.4: Overview of dataset features.

attribute name	type	example	distinct	missing ratio (%)
flight date	datetime	[[2018-08-04 00:00:00, ...]]	1826	0.00
flight number	float	[[1958.0, 202.0, 286.0, ...]]	2036	0.00
tail number	string	[[CSTNP, CSTOP, CSTOD, ...]]	196	0.00
aircraft type code	string	[[320, 332, 343, 319, ...]]	26	0.00
aircraft model	string	[[AIRBUS A320-211, ...]]	26	0.00
fleet	string	[[NB, WB, PGA, WI, nan]]	5	13.01
origin airport	string	[[LIS, EWR, LAD, OPO, ...]]	198	0.00
scheduled departure date	datetime	[[2018-08-04 21:30:00, ...]]	313121	0.00
off-block date	datetime	[[2018-08-04 22:40:00, ...]]	501350	0.00
take-off date	datetime	[[2018-08-04 22:55:00, ...]]	532187	0.19
destination airport	string	[[OPO, LIS, FAO, OUD, ...]]	199	0.00
landing date	datetime	[[2018-08-04 23:27:00, ...]]	529215	0.19
on-block date	datetime	[[2018-08-04 23:36:00, ...]]	516717	0.00
scheduled arrival date	datetime	[[2018-08-04 22:25:00, ...]]	313551	0.00
service type	string	[[J, C, P, G, T, K, F, ...]]	11	0.00
aircraft owner code	string	[[TP, 5K, NI, WI, YU, ...]]	31	0.00
registered delay date	datetime	[[2018-08-04 23:13:59, ...]]	357220	34.64
delay minutes	float	[[42.0, 7.0, 11.0, nan, ...]]	497	34.64
delay code	float	[[16.0, 89.0, 19.0, ...]]	82	34.68
delay sub code	string	[[G, 0, A, nan, C, B, ...]]	20	34.68
ori. air temperature	float	[[34.0, nan, 21.0, ...]]	207	0.84
ori. wind direction	float	[[nan, 290.0, 220.0, ...]]	46	8.73
ori. wind speed	float	[[3.0, 8.0, 5.0, 6.0, ...]]	65	0.59
ori. visibility	float	[[6.21, 10.0, 4.97, ...]]	87	0.25
ori. wind gust	float	[[nan, 23.0, 22.0, ...]]	56	96.26
ori. cloud coverage lvl1	string	[[M, SCT, FEW, BKN, ...]]	11	0.22
ori. cloud coverage lvl2	string	[[M, BKN, SCT, FEW, ...]]	7	0.22
ori. cloud coverage lvl3	string	[[M, BKN, SCT, OVC, ...]]	8	0.22
ori. cloud height lvl1	float	[[nan, 5500.0, 1800.0, ...]]	126	35.94
ori. cloud height lvl2	float	[[nan, 2700.0, 8000.0, ...]]	140	72.46
ori. cloud height lvl3	float	[[nan, 10000.0, 3000.0, ...]]	128	93.93
ori. ice accretion 1h	float	[[nan]]	1	100.00
ori. ice accretion 3h	float	[[nan]]	1	100.00
ori. ice accretion 6h	float	[[nan]]	1	100.00

Continued on next page

Table 3.4 – continued from previous page

attribute name	type	example	distinct	missing ratio (%)
ori. peak wind gust	float	[[nan, 31.0, 30.0, ...]]	15	99.98
ori. peak wind direction	float	[[nan, 230.0, 310.0, ...]]	22	99.98
dest. air temperature	float	[[21.0, 28.0, 31.0, ...]]	219	0.82
dest. wind direction	float	[[0.0, nan, 340.0, ...]]	45	8.72
dest. wind speed	float	[[0.0, 1.0, 9.0, 5.0, ...]]	63	0.60
dest. visibility	float	[[6.21, 4.97, 3.11, ...]]	85	0.24
dest. wind gust	float	[[nan, 22.0, 18.0, ...]]	55	96.26
dest. cloud coverage lvl1	string	[[FEW, M, BKN, SCT, ...]]	11	0.22
dest. cloud coverage lvl2	string	[[M, SCT, BKN, FEW, ...]]	7	0.22
dest. cloud coverage lvl3	string	[[M, SCT, BKN, OVC, ...]]	7	0.22
dest. cloud height lvl1	float	[[4200.0, nan, 1600.0, ...]]	123	36.70
dest. cloud height lvl2	float	[[nan, 5000.0, 12000.0, ...]]	138	72.84
dest. cloud height lvl3	float	[[nan, 10000.0, 3500.0, ...]]	127	93.95
dest. ice accretion 1h	float	[[nan]]	1	100.00
dest. ice accretion 3h	float	[[nan]]	1	100.00
dest. ice accretion 6h	float	[[nan]]	1	100.00
dest. peak wind gust	float	[[nan, 29.0, 28.0, ...]]	16	99.98
dest. peak wind direction	float	[[nan, 270.0, 290.0, ...]]	32	99.98

At first glance, the table shows several attributes with very low filling factors.

3.3.1 Drop attributes without minimum filling factor

As a rule of thumb, when an attribute does not reach a filling factor of 30-40%, dropping it should be considered. A missing factor threshold of 70% was settled, meaning attributes with less representation than 30% were completely removed. This translated in dropping the *ice accretions*, *peak winds*, *wind gust* and *cloud heights levels 2 and 3*.

The number of attributes is now: **52 -> 36**.

3.3.2 Drop instances without delay

The prevalence of missing values in the *delay code* shows that the dataset contains information about flights that met the schedule - observations with flights on schedule, not delayed. This can be seen in the delay distribution in figure 3.2. For this problem, these are not significant and were deleted.

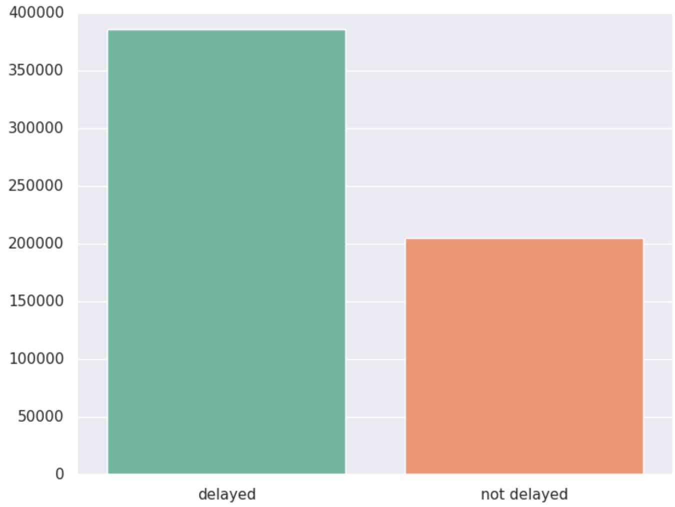


Figure 3.2: Distribution of delayed and not delayed flights in the dataset.

After deletion, the number of observations is now: **591152 ->386153**.

3.3.3 Visualisation by location

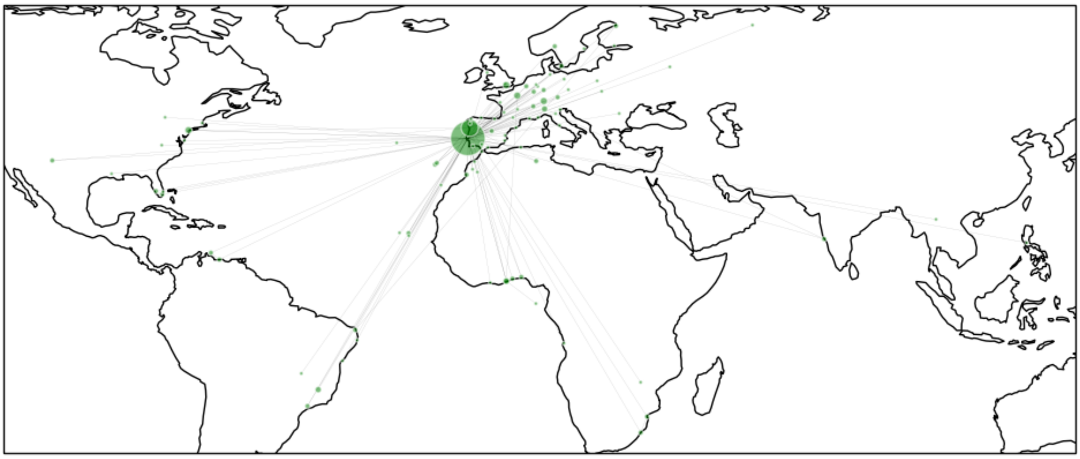


Figure 3.3: Density of flights worldwide per airport. This map was built with data from September of 2017 (94 airports), making use of third-party libraries to extract coordinates from IATA (International Air Transport Association) codes. The dimension of the circles plotted in the map is proportional to the volume of departures and arrivals to the respective airport.

Predictably, since the data regards TAP's activity, the data has a heavy load of flights leaving or entering Lisbon and Porto airports (figure 3.3).

This number is inflated due to 7.5% of the observations being flights between Lisbon and Porto, and 11.5% of all observations being Portuguese domestic flights (flights where departure and arrival take place in the same country [to or from LIS - Lisbon, OPO - Porto, or FNC -

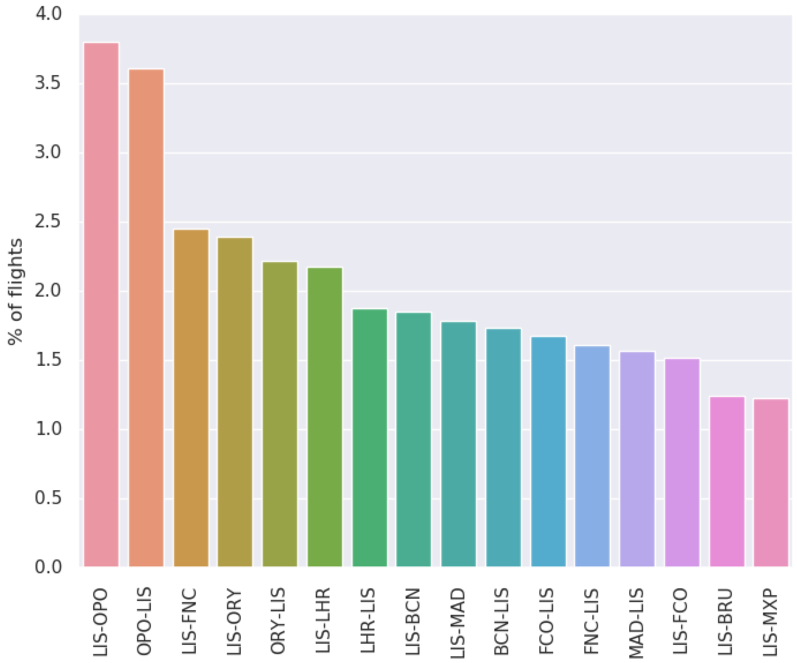


Figure 3.4: Top 16 most frequent connections in percentage from total number of observations.

Madeira]). The majority of the international activity is made in European soil, namely flights between a Portuguese airport and Paris (ORY), London (LHR), Madrid (MAD), Barcelona (BCN), or Rome (FCO). Together, these enumerated European connections factor for 21% of all observations.

In total, there are 188 different airports and 539 unique connections.

The influx of flights is roughly the same for arrivals and departures (figure 3.5) with almost two-thirds of all observations being from or to the airport of Lisbon.

3.3.4 Visualisation by time

From the analysis of the temporal distribution of observations (figure 3.6) it can be seen that the observations in this dataset go from 2013 to 2018, noting that the records from 2013 are only from December's activity and the records from 2018 are from January to November. The distribution per month denotes a slight increase in flights on the holiday season. Unexpectedly, there is no relevant difference between the airline's activity on weekends compared to workdays.

3.3.5 Visualisation by aircraft

As seen in figure 3.7, there are four types of fleets in the dataset (WI, PGA, NB, WB). "WI" is the IATA code for White Airways which mainly operates charter flights. These instances relate to flights hired by TAP for occasional operations. "PGA" is the ICAO (International Civil Aviation Organization) code for Portugália Airlines which is now branded as TAP. Hence, these observations are considered as TAP activity. Also, these airlines only operate with single-aisle aircraft

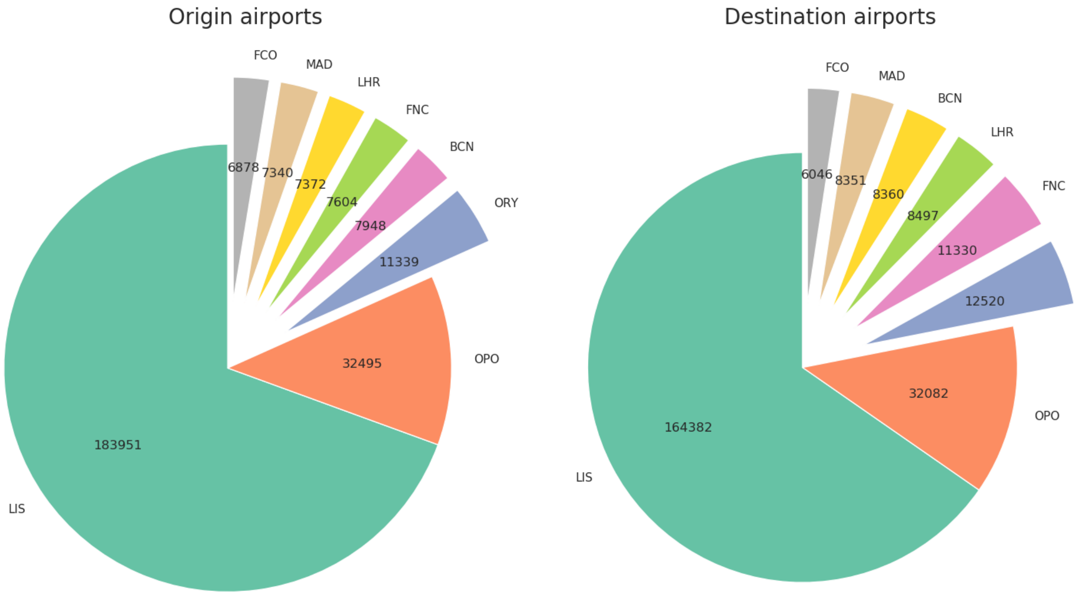


Figure 3.5: Total number of observations of departures and arrivals for the 8 airports with the most observations in the dataset.

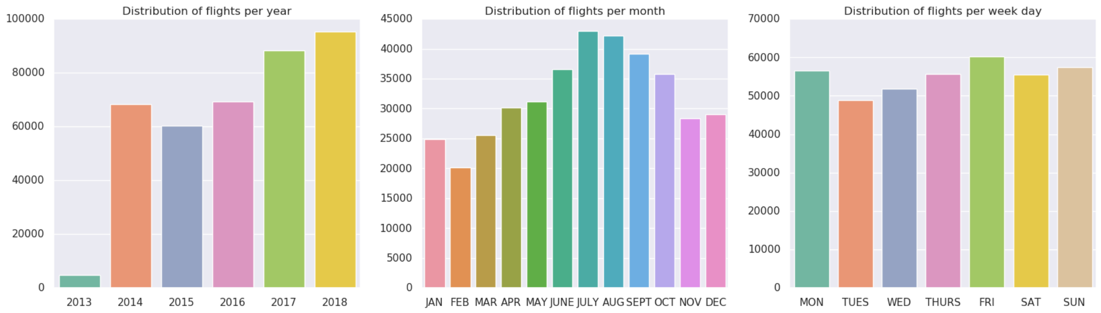


Figure 3.6: Temporal distribution of observations. The first bar plot shows the distribution of observations per year of departure. The second bar plot shows the distribution of observations per month of departure. The third bar plot shows the distribution of observations per weekday of departure.

and, as such, should be considered as part of the TAP's narrow-body fleet. This further contributes to the substantial imbalance shown in figure 3.7.

Fleet adjustments The *fleet* feature has some missing values, as seen in table 3.4. Since the fleet refers mainly to the aircraft's fuselage size, this can be fixed with the help of a domain expert, using the aircraft's model as a reference.

There is also a strong relationship between the fleet type and the expected duration of the flight. In aviation, short-haul flights go from 30 minutes to 3 hours, medium-haul flights go from 3 to 6 hours, and long-haul flights take longer than 6 hours. Almost all short and medium-haul flights are

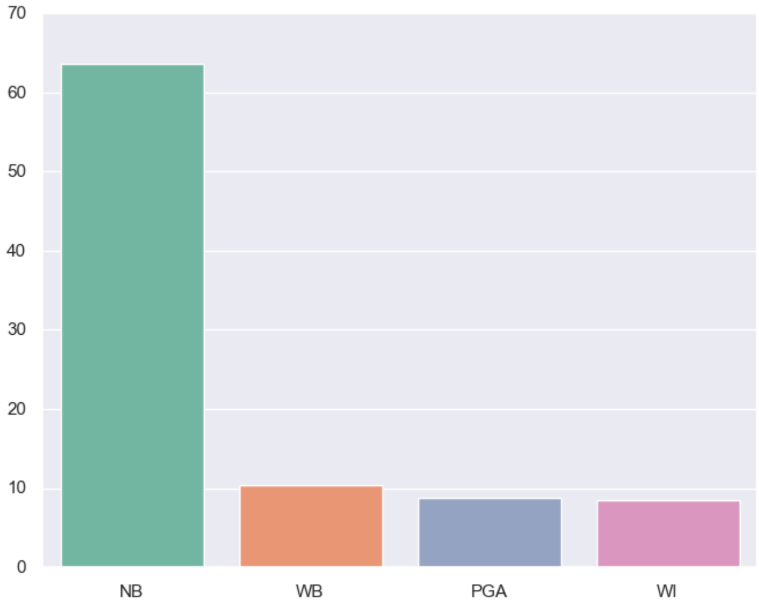


Figure 3.7: Percentage of observations for each *fleet* category.

made with narrow-body aircraft while long-haul flights are made with wide-body aircraft (figure 3.8).

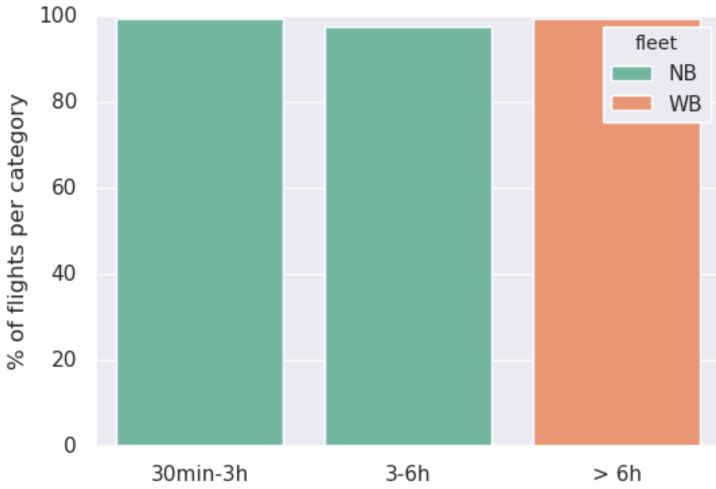


Figure 3.8: Percentage of observations of short-haul, medium-haul and long-haul flights and the predominant *fleet* category for each one.

The fleet imbalance of observation is further analysed through figure 3.9, showing that most of the observations are made by two types of aircraft only.

Figure 3.10 displays the percentage of flights each unique aircraft made (unique tail number) and the respective aircraft fuselage category. The wide-body aircraft with a higher number of flights made is the 68th aircraft with most flights made.

This imbalance can lead the models to develop a bias towards the majority class (in this case,

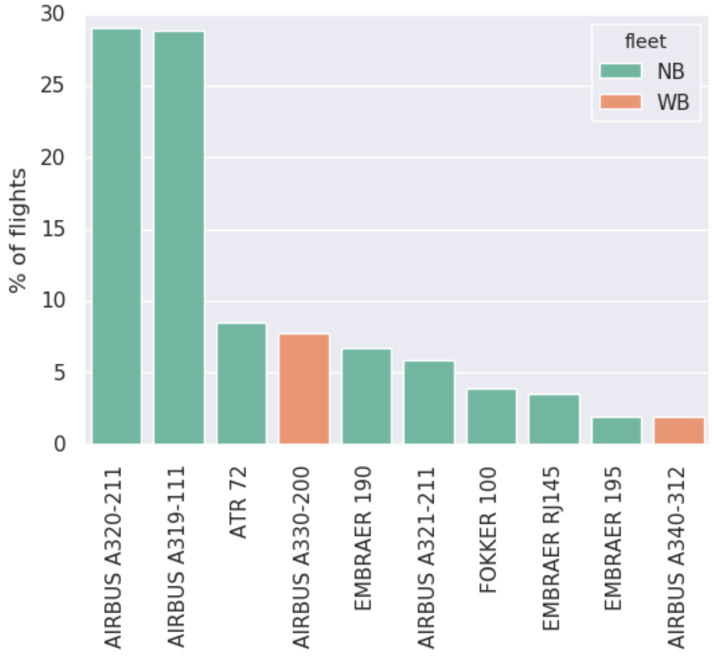


Figure 3.9: Percentage of observations from the 10 most prevalent *aircraft models* and respective *fleet* category.

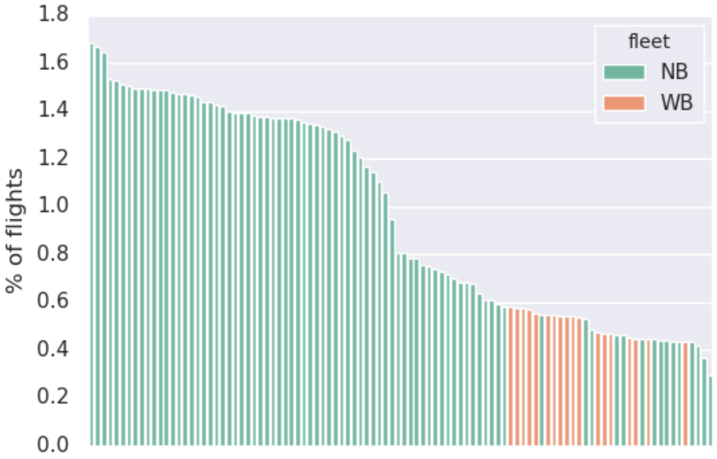


Figure 3.10: Percentage of observations from the 100 most used aircrafts and their respective *fleet* category.

narrow-body flights) and acquire no predictive capabilities for the minority class (long-distance flights). The latter having much larger absolute values in the target variables may as well drastically raise the overall error metrics.

3.3.6 Visualisation by delay

Figure 3.11 shows a reduction in delay at the last hours of the day but an increase in the average delay minutes for the same period. This can be further explained in figure 3.12. For that same after-hour period there is a thinning in the number of flights. Consequently, the reduction in the absolute value of delayed minutes per hour is due to the decrease in activity in those hours.

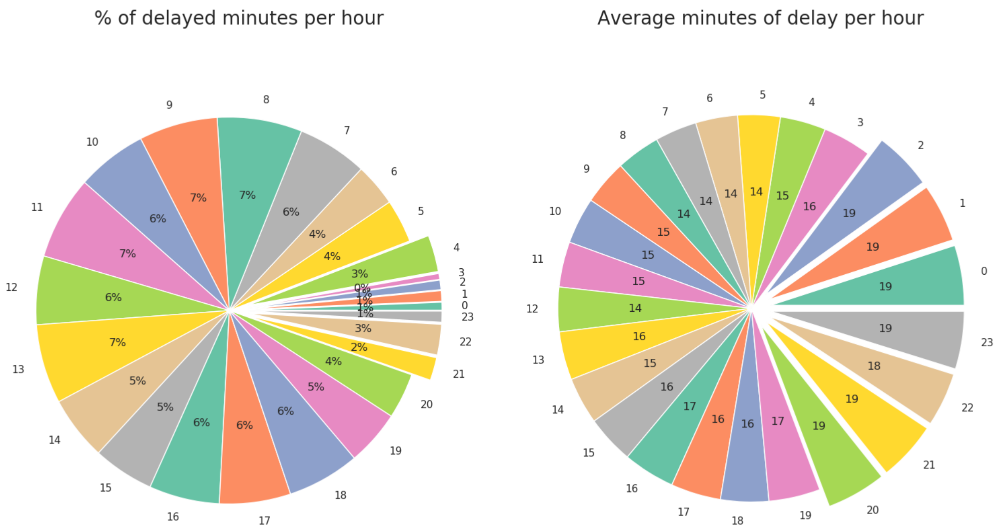


Figure 3.11: Correlation between the hour of the day and departure delay. The first pie chart shows, for each hour, the percentage of delayed minutes from the total amount of minutes flown. The second pie chart shows, for each hour, the absolute value of the average minutes of delay.

Also, there are several significant delays throughout one single day.

In aviation, an airline departure or arrival, which is considered to be on time has a departure or arrival that occurs within 15 minutes of the scheduled time [WZHC19]. As can be seen in figure 3.12, the more prevalent delay times are within [30, 60] minutes, that is further supported by the work in [AL15], which was developed for the air traffic in one of the most represented airports in this dataset (i.e. OPO airport).

Delay codes from 81 to 84 regard restrictions on the management of air traffic flow, from 85 to 89 are airport and governmental authorities, and from 91 to 96 are reactionary. The most prominent delays (from figure 3.13) are "restrictions at airport of departure" (89), "aircraft rotation, late arrival of aircraft from another flight or previous sector" (93), "ATFM (Air Traffic Flow Management) due to restriction at destination airport" (83), "ATFM due to ATC en-route demand" (81) and "ATFM due to weather at destination" (84) [EUR17, EUR18]. Interesting to note a great number of reactionary delay, especially in Madeira's airport. Also, no considerable impact of weather on Lisbon's airport activities (important because of the huge representation of Lisbon's airport in the dataset).

The *delay sub code* represents a more detailed description of the originating problem. Despite being represented the same way for different *delay codes*, *delay sub codes* do not have a standalone representation and should always be preceded by a *delay code*.

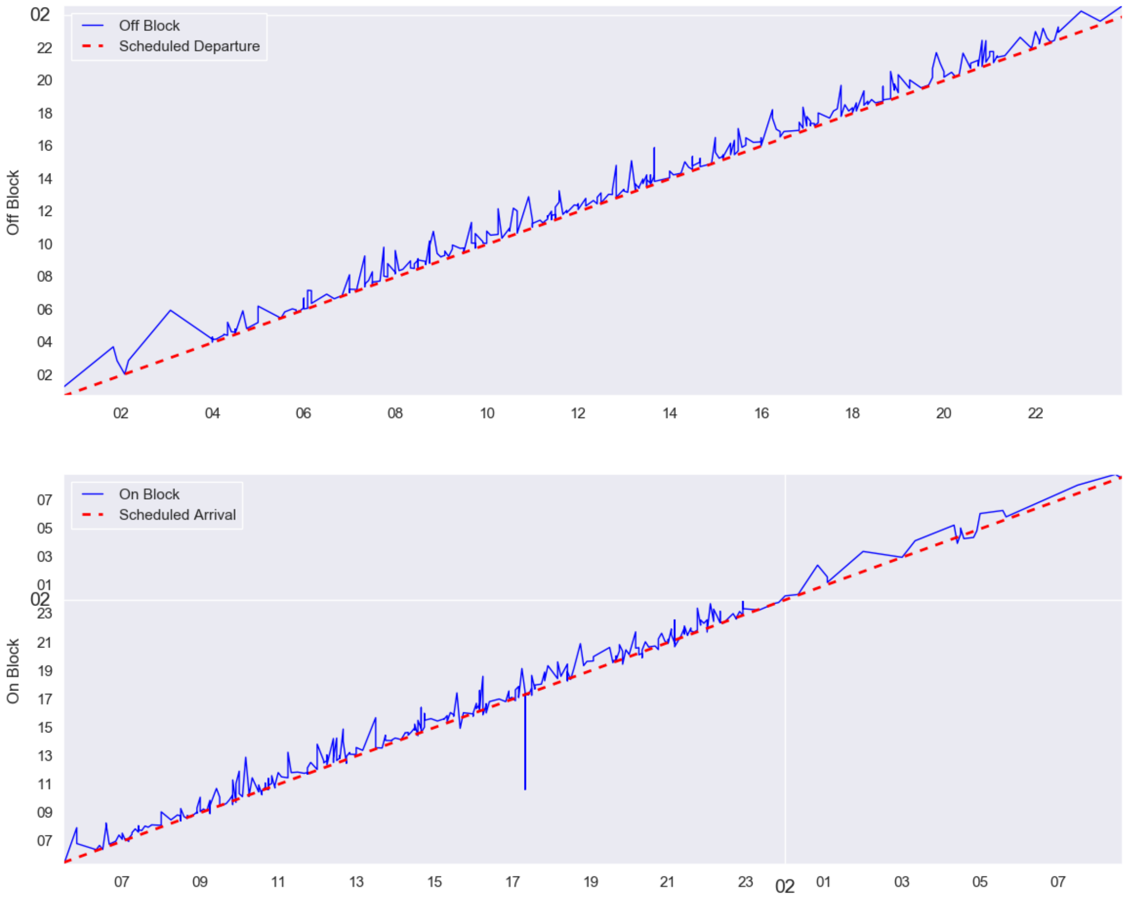


Figure 3.12: Correlation between scheduled and actual times for data from September of 2017. The first plot shows the actual *off-block* times against the respective *scheduled departures*. The second plot shows the actual *on-block* times against the respective *scheduled arrivals* (for the same observations as the first plot).

Build new delay code Having no value on its own, it is wise to concatenate the *delay sub codes* to the respective *delay codes*. This concatenation generates alphanumeric values with a leading number (*delay code*) and a trailing character (*delay sub code*). After this process, the *delay sub codes* were removed.

The number of attributes is now: **36 -> 35**.

3.3.7 Drop badly formed instances

In the second plot of figure 3.12, an outlier is clearly showing. That outlier has null *landing* and *take-off* values because it is a training flight instance. There are also instances of flights that were cancelled prior to the aircraft taking off (e.g. due to bad weather) or had values that did not respect the sequencing between the flight times (e.g. *take-off date* after *on-block date*) due to measurement errors. There were 1277 instances with the same origin and destination airport and one instance where the *landing date* was greater than the *on-block date*.

After deletion, the number of observations is now: **386153 ->384875**.

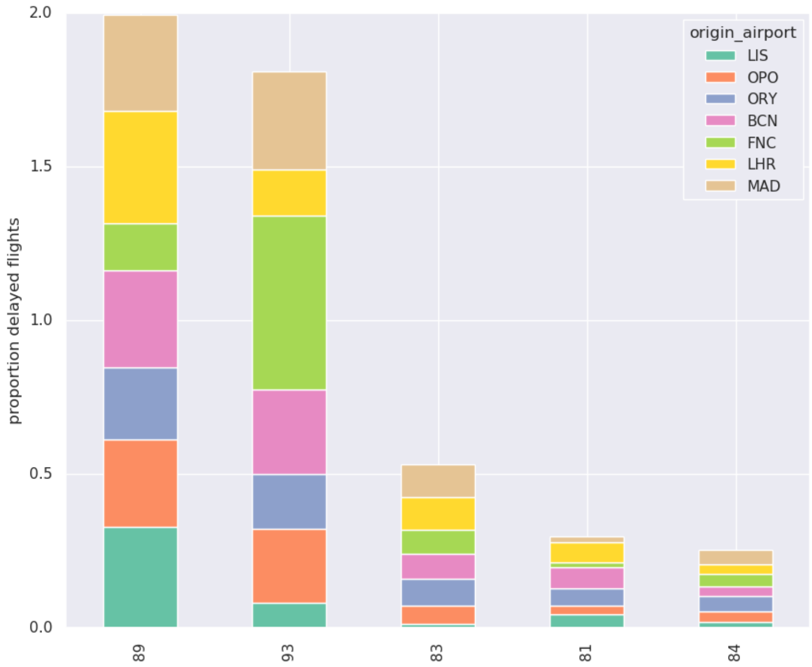


Figure 3.13: Proportion of observations of the seven most represented airports for the 5 most represented *delay codes*.

3.3.8 Visualisation by weather

Weather data on origin and destination is quite similar, as the graphs can be seen to overlap in figure 3.14. The distribution of temperature does not show temperatures that could lead to severe ice accumulation. There is a prevalence of winds coming from the northwest. Wind speed, visibility and cloud height have outliers that may require removal.

According to weather data provider's documentation [Uni], missing values on categorical data take the return value 'M', also seen in figure 3.15, making levels 2 and 3 not reach the filling threshold of 30% and should, therefore, be removed.

The number of attributes is now: **35 -> 31**.

There does not seem to be any discernible pattern in the distribution of cloud height during a whole month with an even scatter of points (figure 3.16). Most instances have cloud coverage of "FEW". This behaviour can be assumed to extend throughout data since the dataset is filled with activity in Lisbon's airport and the plotted info is from the representative month.

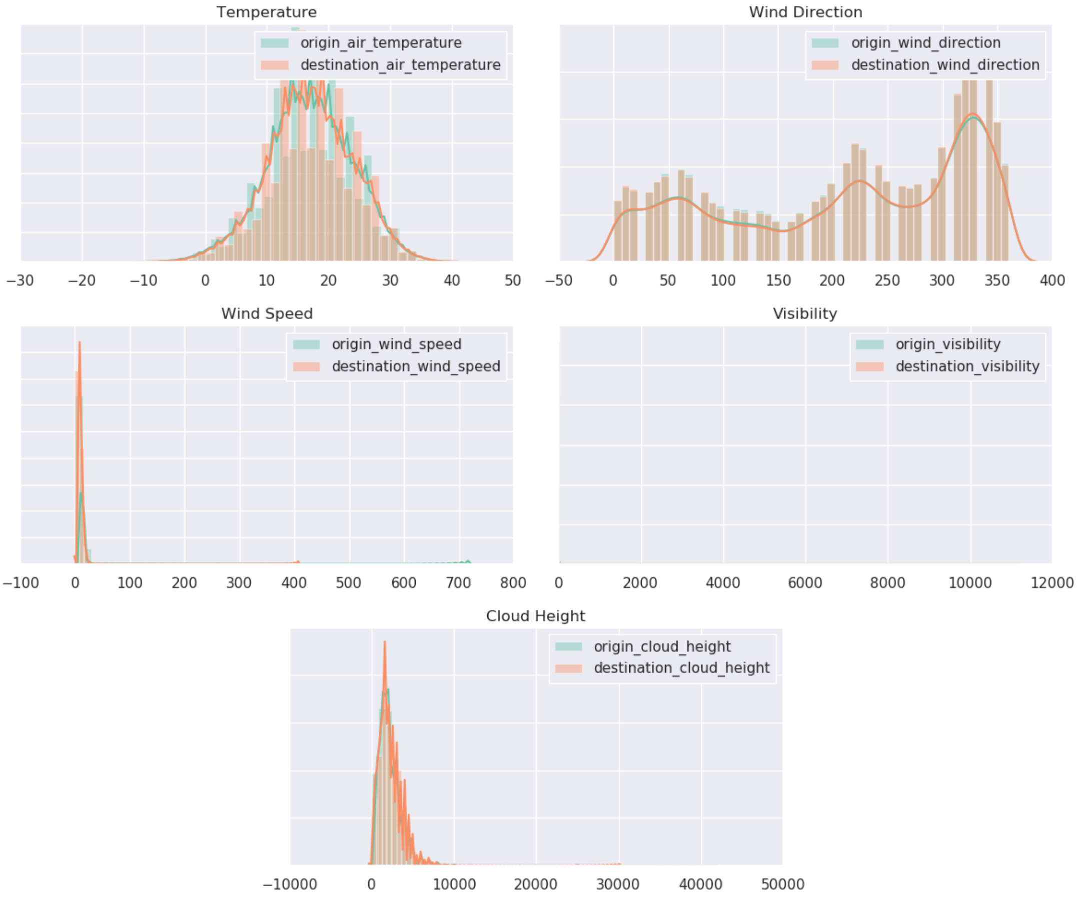


Figure 3.14: Value distribution of numerical weather features. The forth histogram regard *visibility* but has a single huge outlier which makes the plot indiscernible.

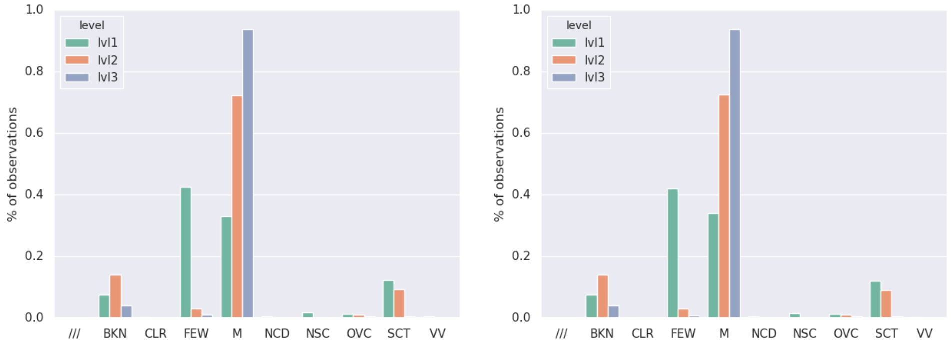


Figure 3.15: Percentage of observations for each *cloud coverage* label and level. The first bar plot using data from the origin airport, and the second bar plot using data from the destination airport.

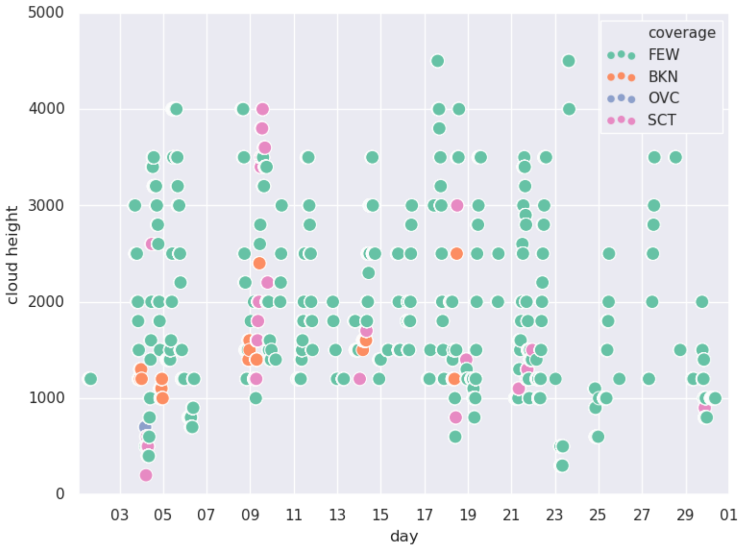


Figure 3.16: Cloud height and density at Lisbon's airport in September of 2017.

Chapter 4

Feature Engineering

Feature engineering consists in using domain knowledge, feature extraction and visualisation techniques to enhance data in order to make complex structures in data more perceivable by learning algorithms. This section describes the application of these procedures to the dataset used in this work.

4.1 Data Enhancement

Enhancing features can make the information more easily modelled and avoid overfitting models.

4.1.1 Outlier Removal

Figure 3.14, that presents the value distribution of numerical weather features, shows big distribution tails which means the existence of extreme deviations (or outliers). Because these observations deviate markedly from the distribution, it is safe to assume these are erroneous and should be deleted.

- *origin wind speed* has a maximum value of 721 and 142 values greater than 30 for a mean value of 8.11.

- *destination wind speed* has a maximum value of 409 and 131 values greater than 30 for a mean value of 8.06.

- *origin cloud height* has a maximum value of 41700 and 411 values greater than 10000 for a mean value of 2128.33. Also, 722 observations with clouds at 0 feet.

- *destination cloud height* has a maximum value of 30000 and 492 values greater than 10000 for a mean value of 2192.18. Also, 530 observations with clouds at 0 feet.

- *destination visibility* has a maximum value of 11190.9 and 1 value greater than 100 for a mean value of 6.02. Also, three observations with visibility 0 and mismatching cloud density.

The outlier thresholds, or the values that distinguish whether an observation is an outlier or not, were set by visual reference through plot analysis.

After deletion, the number of observations is now: **384875 ->382444**.

4.1.2 Wind Direction Binning

Wind direction is measured in degrees and, as such, it should be assumed that it is a continuous variable, but the distribution makes it seem discrete. Binning may throw away unwanted granularity and makes it more interpretable. Thus, the variable was discretised up to a secondary intercardinal direction. Meaning, it now has 16 unique labels - the cardinal, the intercardinal and the secondary intercardinal directions.

4.2 Create new features

Data features directly influence predictive performance and achieving results. Unlike deep learning architectures, most machine learning algorithms are heavily dependent on the quality of those features. The goal is to create features that describe the complex structures in data and make patterns more visible to the learning algorithms.

4.2.1 Target creation

The data does not contain the target attributes, although they can be inferred using the current information from the flight time decomposition 3.1. The target attributes were calculated as follows:

- ***air time*** = *landing date* - *take-off date*
- ***taxi out*** = *take-off date* - *off-block date*
- ***taxi in*** = *on-block date* - *landing date*
- ***actual block time*** = *on-block date* - *off-block date*

The number of attributes is now: **31 -> 35**.

4.2.2 Create scheduled block time

The scheduled block time is the total amount of time the airline was expecting the flight to take. This variable is more verbose than the decomposition of scheduled departure and arrival dates. The targets will probably have a high correlation with this variable since they increase and decrease in a similar proportion. The value is computed as the difference between the scheduled arrival and departure dates.

The number of attributes is now: **35 -> 36**.

4.2.3 Create night departure attribute

As seen in figure 3.11, the flights behave differently in the after-hours period. Through plot analysis, this interval was approximated to the interval between 21 and 3 o'clock. It might be interesting to compile a boolean value that distinguishes high from the low activity period. This variable takes the truth value of the condition where the *scheduled departure date* is within the interval from 21 to 3 inclusive.

The number of attributes is now: **36 -> 37**.

4.2.4 Create scheduled rotation time

Scheduled rotation time is the elapsed time between *scheduled departure date* and previous leg/flight *scheduled arrival date*. It may be able to map the time needed for maintenance, cargo loading, refuelling, boarding, and any other ground action.

In some scenarios, these features could be calculated using *on-block time*, but this would set strict constraints on the developed model's usage. The prediction may be made within 24 hours but having *scheduled rotation time* data computed with *on-block time* instead of *scheduled arrival date* would imply that the aircraft subject to evaluation could not be airborne. Meaning, even if a reactionary delay were detected a few legs prior, the *on-block time* from the previous leg would not be known until that flight had been concluded.

The number of attributes is now: **37 -> 38**.

4.2.5 Create previous delay code

While the *delay code* is not known until the flight arrives, the delay of the previous leg may be a good predictor of the following one. In reactionary delays, the same *delay code* is propagated forward several legs which might even create a high correlation between *delay code* and *previous delay code*.

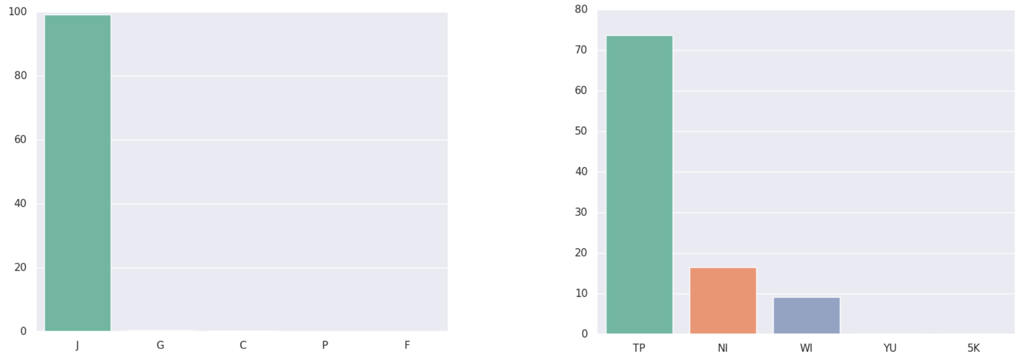
The assessment of the previous flight is made grouping observations by *tail number* and sorting them chronologically (using *scheduled departure date*).

With the creation of *previous delay code* and removal of *delay code*, the number of attributes remains the same: **38 -> 38**.

4.2.6 Date unfolding

Dates are not a suitable format for most modelling algorithms. So, the *scheduled departure date* and *scheduled arrival date* were split into five new features each - year, month, day, hour, and minute. It would be possible to have a sixth, weekdays, but seems unnecessary since data analysis did not capture any trend based on the weekdays.

The number of attributes is now: **38 -> 46**.



(a) Service type distribution

(b) Aircraft owner distribution

Figure 4.1: Service type and aircraft owner distributions.

4.3 Drop unnecessary features

Finding and removing redundant attributes can significantly minimise the training time and contribute to avoiding the curse of dimensionality. Fewer attributes are generally better, as long as those few attributes contain relevant information for the model do properly generalise the data and not underfit. However, knowing if the data collected is necessary for the task at hand is an intrinsic uncertainty of every data mining problem. Whether it is because of lack of representation for the minority classes or because it does not conform the scope of the project, yet more attributes and observations were selected for removal.

Figure 4.1a shows that *service type* has a minimal representation of all classes but one, so only flights that are of type "J" (Scheduled Passenger Service) should be considered. 4.1b represents a feature that brings no value to the predictive task. In this case, all instances from this dataset are from TAP's activity, so whether or not TAP owns the aircraft, these are scheduled as others are.

After deleting the "non-J" records, the number of observations is now: **382444 ->379599**.

The dropped attributes were:

- redundant features:
  - *flight date* - contains a subset of the information provided by *scheduled departure date*
  - *aircraft type code* - is a coded version of the *aircraft model*
- do not bring useful information:
  - *aircraft owner* - as seen in figure 4.1b, aircraft ownership would bias the information
  - *service type* - only one unique value now
- unfit for project
  - *registered delay date*
  - *delay minutes*
  - *off-block date*

- *take-off date*
- *landing date*
- *on-block date*

After feature dropping, the number of attributes is now: **46 -> 36**.

4.4 Feature Selection

Often, real-world data is noisy and may contain features that do not promote an increase in the model's performance. The idea behind feature selection is to study this relation and select only the variables that show a strong correlation. Doing so may have several benefits such as faster training time, decrease in complexity, improved performance or reduction of over-fit.

4.4.1 Correlation Matrix

Correlation matrices are especially useful to assess the correlation between numerical variables.

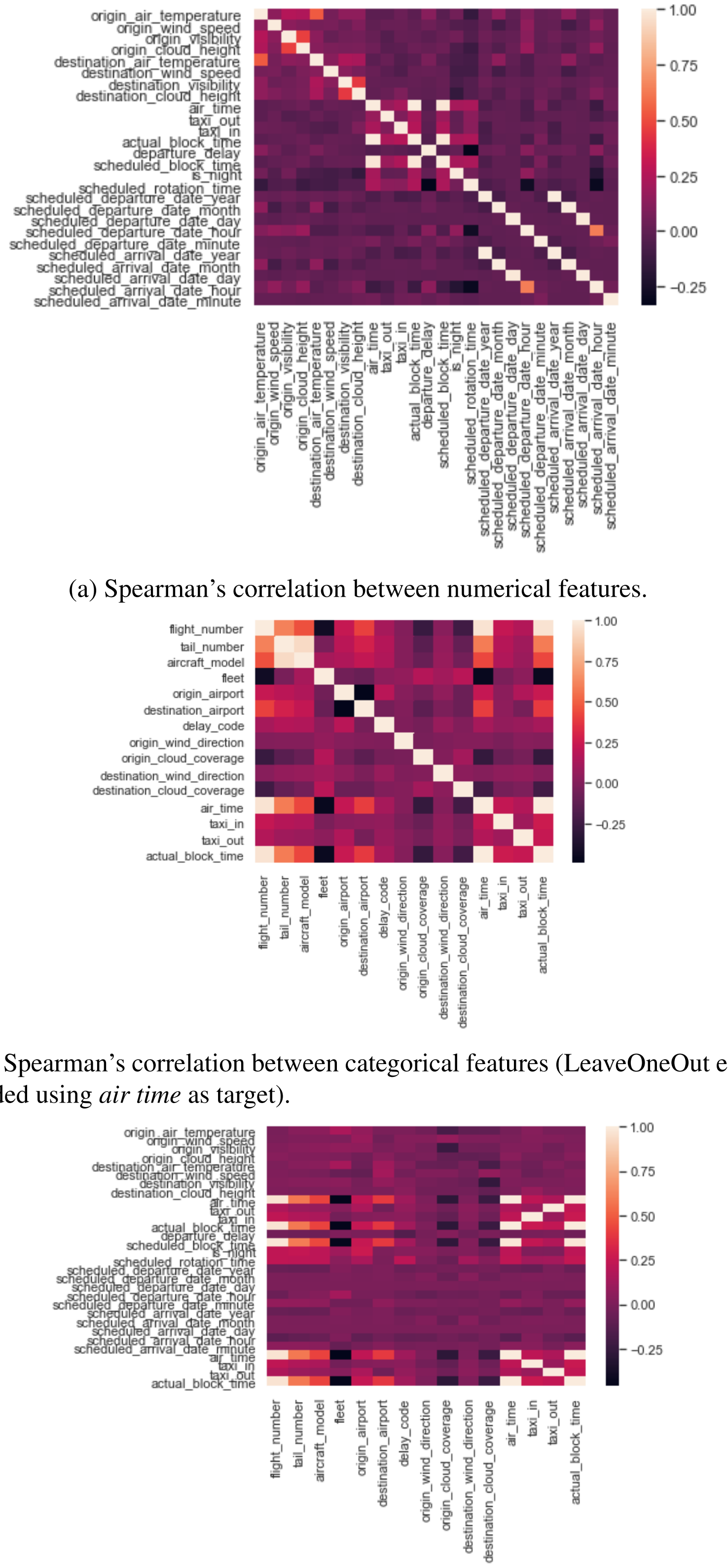
The rank correlation measure used is Spearman's rank correlation coefficient because it assesses monotonic relationships which can express non-linear relationships. Also, as previously seen by analysing the distributions of the numerical variables, they are neither homoscedastic nor normally distributed, which would otherwise be assumed by Pearson's correlation. Correlation heatmaps are useful to assess multicollinearity which is an issue that might make difficult for the learning algorithms to estimate the relationships between predictors and target. If multiple predictors change in unison (multicollinearity), estimated coefficients become very sensible to small changes. This behaviour is particularly prominent in linear models.

Figure 4.2a shows that the year, month, and day of departure are often the same as those of arrival. From figure 4.2b, it can be perceived that that *tail number* and *aircraft model* are highly correlated. Analysing figure 4.2c, *flight number* is highly correlated with *scheduled block time*. All these relationships express multicollinearity and may lead to modelling issues.

4.4.2 Data Compression

Data compression may be useful to lower data dimensionality and avoid multicollinearity. One of the most used tools for this purpose in machine learning is Principal Components Analysis (PCA). PCA is a statistical procedure that converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables, called principal components. Each of these components explains part of the data's variance.

PCA is a lossy compression algorithm, so compressed data will not be the same as the original data. Nevertheless, principal components can help models reach similar results quicker than using the original data.



(a) Spearman's correlation between numerical features.

(b) Spearman's correlation between categorical features (LeaveOneOut encoded using *air time* as target).

(c) Spearman's correlation between numerical and categorical features (LeaveOneOut encoded using *air time* as target).

Figure 4.2: Heatmaps of spearman's correlation between features.

PCA is sensitive to the relative scaling of the original variables and works only for numerical variables. Meaning, it requires categorical encoding and feature scaling (standardisation because it assumes zero-centric data).

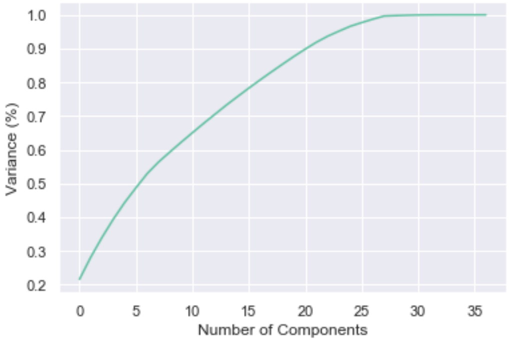


Figure 4.3: Plot of the cumulative summation of the explained variance.

Figure 4.3 shows that 27 principal components explain almost 100% of the data.

4.5 Imputation

Missing values are common in any machine learning task. Several reasons may lead to missing values, and these affect the performance of learning algorithms. Most ML algorithms do not accept datasets with missing values. So the dilemma is whether to impute or remove the data.

4.5.1 Types of Missing Data

There are different types of missing data, depending on why it went missing. Each type has different constraints on how it should be handled.

- **Missing Completely at Random (MCAR)**: There is no relationship between the missingness of the data and any values, observed or missing. When data is missing completely at random, it means that the records with missing data can be safely removed and undertake the analyses using only observations that have complete data.

- **Missing at Random (MAR)**: The propensity for a data point to be missing is not related to the missing data, but it is related to some of the observed data. It is assumed a probabilistic relationship between the missing data and observed data, which means other features can predict data MAR in the dataset.

- **Missing not at Random (MNAR)**: There is a relationship between the propensity of a value to be missing and its hypothetical value. The missingness is related to factors that were not accounted for (e.g. people with high salaries generally do not reveal their incomes in surveys). MNAR is called "non-ignorable" because the missing data mechanism itself has to be modelled.

In the first case, it is safe to remove the data with missing values. In the second case, it can be removed or imputed (imputation does not necessarily give better results). In the third case, removing observations with missing values can produce a bias in the model, so it is unadvised.

4.5.2 Dataset Missing Data

Currently, the distribution of missing values stands as illustrated in figure 4.4. Previously, the most prominent missing attributes were deleted, but some variables still have substantial missing values. Since these remaining variables do not go over 40% missing values, it seems feasible to use imputation. The missing data is primarily found in the weather data. There is no apparent reason for this, so the data is regarded as MAR.

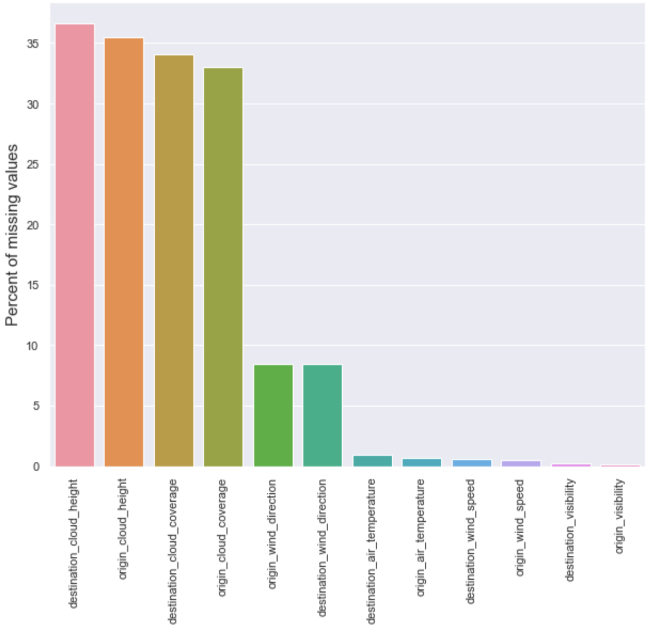


Figure 4.4: Percentage of missing data by missing feature.

4.5.3 Dataset features

As seen in figure 4.4, the features with missing data are weather features, so it seems appropriate to drop aircraft and flight specific features since there is intuitively no correlation between these values and meteorological behaviour. Meaning, the following features were dropped for the imputation task: *flight number*, *tail number*, *aircraft model*, *fleet*, *taxi out*, *air time*, *taxi in*, *actual block time*, *scheduled block time*, *scheduled rotation time*, *previous delay code*. The remaining data is composed of weather data and location-based attributes.

The number of attributes for imputation is now: **36 -> 25**.

4.5.4 Input-output creation

To train a predictive model to generalise for unobserved data its required a testing environment capable of mimicking the real conditions. In this scenario, it is required an input set with missing values and its counterpart with the actual values.

To build to input data, firstly, only observations with no missing values are kept. Then, a fraction of the data (50% [arbitrarily set]) is subject to missing value imputation (just weather features). This imputation is done for a ratio of 40% of missing values (matching the highest missing factor in 4.4). This process creates an input set with missing values (roughly 40% in the weather features), similar to the actual ratio of missing values. The complementary output set is that same fraction of the data without the missing value imputation.

4.5.5 Imputation data preprocessing

The use of predictive models generally requires data preprocessing, such as categorical encoding, train-test split and feature scaling.

Categorical encoding

For the imputation problem, the encoding needs to be reversible to assess the performance of the models and use the newly imputed data. Both label and one-hot encoders have trivial decoding but, through testing, one-hot encoding achieves better performance than label encoding. The drawback of one-hot is a drastic increase in the dataset's dimensionality.

After encoding, the number of attributes is now: **25 -> 294**.

One-hot encoding this data required extra work to suit the problem. The imputed missing values need to be propagated to every other subcolumn of the particular feature in an observation. Also, the existence of missing values in the training data results in less unique labels and, consequently, will generate fewer subcolumns than on the test data. Train and test sets need to have the same shape, so the missing subcolumns should be added as well (zero-filled).

Feature scaling

Bringing values to the same magnitude is important to avoid features with greater absolute values having more significance than features with lower absolute values.

The scaling technique used was min-max normalisation, which consists of rescaling the range of features to be within the range of [0, 1]. Scaling is required because models cannot interpret missing values, so these have to be encoded in some way. In this case, as -1. Because *air temperature* reaches negative values, rescaling data between 0 and 1 emphasises the missing values and makes the model converge faster.

The scaler is fit in the observations without missing values from the training set, and all training, validation and test inputs and outputs are normalised with that scaler.

4.5.6 Autoencoder solution

Autoencoders may be able to handle the missing value problem. The following approach was motivated by the implementations of autoencoders for image denoising, as in [VLBM08], where the input contains noise (in this case, missing values), and the output does not.

Brief introduction

Autoencoders are a specific type of feedforward neural networks where the input is the same as the output. They are comprised of two networks: an encoder and a decoder. They compress the input into a lower-dimensional representation (encoding) and then reconstruct the output from this representation (figure 4.5).

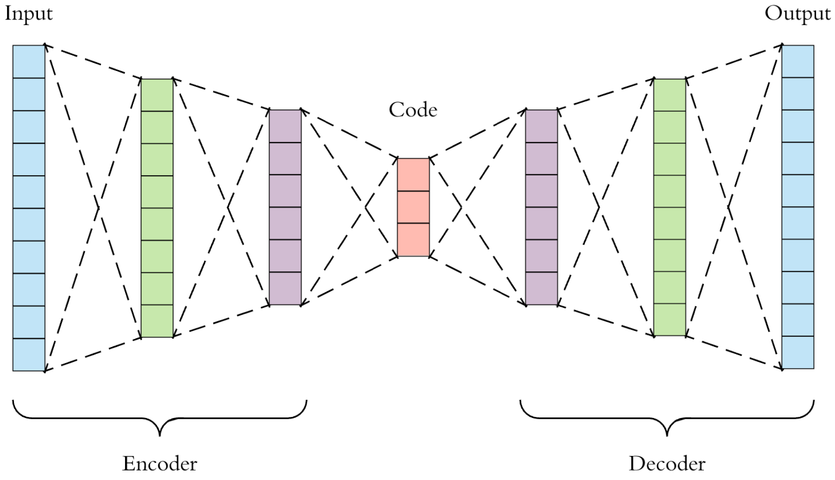


Figure 4.5: High level illustration of a regular autoencoder.

Both the encoder and decoder are fully-connected feedforward neural networks. Typically, the entire network is trained as a whole, optimised via back-propagation.

The loss function, known in this context as the reconstruction loss, penalises the network for creating outputs different from the respective inputs.

As the latent vector (the output of the hidden layer in the middle, the bottleneck of the network or latent space) has far fewer units than the input, the encoder must choose to discard some information. The encoder learns to preserve as much of the relevant information as possible in its output, discarding unnecessary parts. The decoder learns to take the encoding and accurately reconstruct it into a full input-like sample. Together, they form an autoencoder.

Setup

The used architecture has three hidden layers. The outer ones with size 250 and the inner one with size 100. Every hidden layer has a dropout regularisation (avoid over-fitting) [SHK^{+}14] of 0.2 and is initialised with Xavier initialisation (more robust to exploding gradients) [GB10]. Every layer is activated with a ReLu function besides the output one, activated with the sigmoid function. The loss function used was MSE, and the optimiser was Adam.

This setup is the best from a few experiments.

In training, early stopping was used with patience of 5 and min delta of 5e4, meaning, it imposes a stopping rule were the error must be lower in a given iteration than its five previous ones, for a difference of at least 5e4.

Masked Autoencoder

After testing the standard autoencoder, a slight improvement was tested using a mask alongside the input data to address only the missing cells when computing the loss function. This network's input is the concatenation of a mini-batch matrix from the dataset with missing data (X) and the corresponding missing value boolean mask ($M = \{(i, j) \mid z_{i,j} \neq \varnothing\}$). This is given by the augmented matrix $[X_{(i\times j)}M_{(i\times j)}]$ where i is the batch size and j the number of attributes. This approach is made for the mask to be accessible, at run-time, to the custom loss function. The loss function computes the MSE just for the fields with empty values.

Figure 4.6 illustrates the employed architecture. The best network setup was achieved five hidden layers, the outer ones with four times the input shape, the inner ones with two times the input shape, and the centre one (the latent space) with input shape. All layers have ReLU as their activation function besides the output one that uses sigmoid. Also, all layers have their weights initialised with Xavier distribution and have a dropout of 20%.

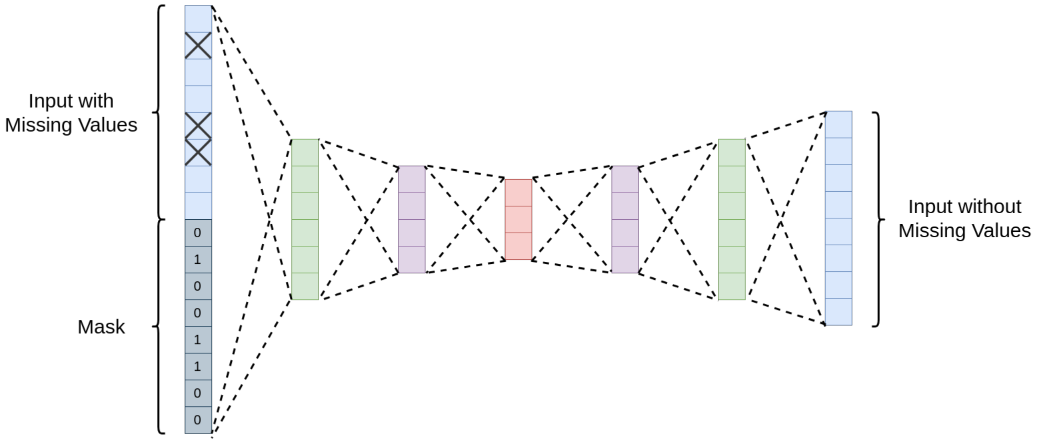


Figure 4.6: Illustration of the masked Autoencoder architecture

4.5.7 Variational Autoencoder

Another approach that could address the problem would be to generate synthetic values from a generative model, namely, a Variational Autoencoder (VAE) [KW19]. If a good reconstruction error is achieved, the missing values can be sampled from the autoencoder.

Brief introduction to VAE

The fundamental problem with autoencoders, for generation, is that the latent space they convert their inputs to and where their encoded vectors lie, may not be continuous, or allow easy interpolation. If the latent space has discontinuities, the decoder will not be able to properly reconstruct data because, during training, it never saw encoded vectors coming from that region of latent space.

Variational Autoencoders, on the other hand, have a unique property that their latent spaces are, by design, continuous, allowing easy random sampling and interpolation. It achieves this by making its encoder output two vectors: a vector of means, μ, and another vector of standard deviations, σ. These vectors are then used to sample from a Gaussian distribution and pass the sampled output onwards to the decoder.

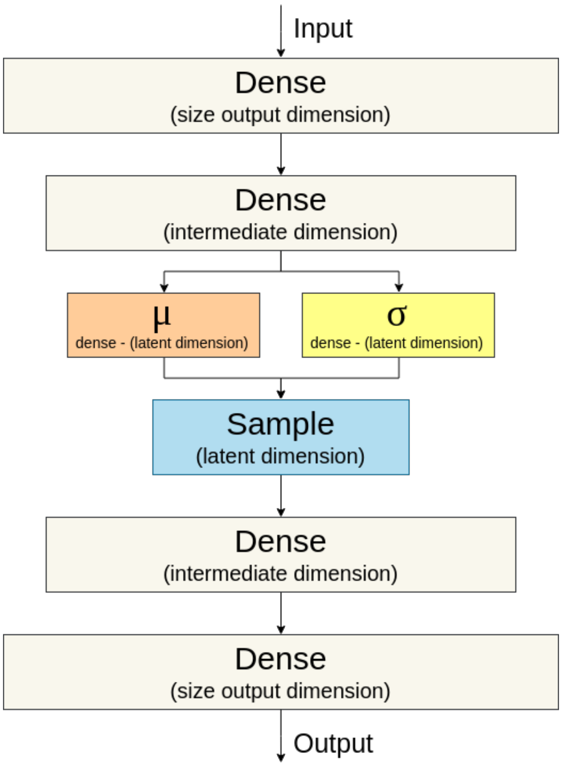


Figure 4.7: Illustration of a Variational Autoencoder architecture.

This stochastic generation means, that even for the same input, while the mean and standard deviations remain the same, the actual encoding will somewhat vary on every single pass.

The decoder learns to match, not only a single point in latent space but all nearby points (region centred at the mean with the radius as standard deviation) to a point in the output. This property allows the decoder not just to decode single, specific encodings in the latent space (leaving the decodable latent space discontinuous), but ones that slightly vary too, as the decoder is exposed to a range of variations of the encoding of the same input during training.

To force encodings to be as close as possible to each other while still being distinct, allowing smooth interpolation, and enabling the construction of new samples, VAEs use the Kullback–Leibler divergence (KL divergence) into the loss function. The KL divergence measures how much two probability distributions differ from each other. Minimising the KL divergence means optimising the probability distribution parameters (μ and σ) to resemble that of the target distribution.

Setup

The approach used was the same as in [MKA18]. The results in tables 4.3, 4.1 and 4.2 were achieved with a 5 hidden layer architecture. The outer layers with size 64, the inner layers with size 32 and the "code" with size 16. The weights were initialised with Xavier initialisation and gradient clipping used to address an exploding gradient problem in training. Also, early-stop was used to avoid overfitting.

4.5.8 Result Comparison

A baseline model was created to assess the viability of applying the resulting outputs. The baseline's results were computed using the mean values for each numerical feature and the most common values for each categorical feature.

The results are assessed by computing the residuals between the predicted and actual values just for the missing valued cells.

Table 4.1: Comparison of imputation performance for each numerical feature using different autoencoders.

Attributes	Autoencoder			Variational Autoencoder			Baseline		
	MAE	MSE	MAPE*	MAE	MSE	MAPE	MAE	MSE	MAPE
origin air temperature	3.878	26.158	-	6.511	69.089	-	4.819	37.979	-
origin wind speed	3.396	18.196	-	4.685	36.362	-	3.591	20.835	-
origin visibility	0.641	1.274	49.531	0.727	2.452	53.340	0.671	1.360	45.840
origin cloud height	9.17e2	1.40e6	1.07e2	1.31e3	2.94e6	1.42e2	9.93e2	1.68e6	1.18e2
destination air temperature	5.335	43.733	-	6.707	72.822	-	4.978	40.068	-
destination wind speed	3.415	18.492	-	4.717	36.386	-	3.597	20.989	-
destination visibility	0.592	1.170	32.659	0.635	2.109	31.976	0.589	1.204	35.099
destination cloud height	9.15e2	1.45e6	90.331	1.33e3	3.08e6	1.26e2	1.01e3	1.74e6	1.03e2

* MAPE does not work for attributes with observed values of zero

Analysing table 4.1 it can be seen that the autoencoder have particular trouble predicting cloud heights. With an error of almost double the actual value (MAPE of 100%). Also, these are the features with the highest absolute value that consequently massively inflate the overall performance of the models.

The superior predictive capacity of the model to accurately assess *cloud coverage* values when compared to *wind direction* values is probably due to the lower number of distinct labels of the former when compared to the latter.

Table 4.2: Comparison of imputation performance for each categorical feature using different autoencoders.

	Autoencoder	Variational Autoencoder	Baseline
origin wind direction	0.124	0.083	0.122
origin cloud coverage	0.670	0.532	0.666
destination wind direction	0.149	0.081	0.116
destination cloud coverage	0.662	0.526	0.659

Table 4.3: Overall performance of the different autoencoders on the imputation task.

	Autoencoder	Masked Autoencoder	Variational Autoencoder	Baseline
MAE	2.35e2	2.31e2	3.32e2	2.51e2
MSE	3.54e5	3.46e5	7.50e5	4.14e5
Accuracy	0.402	0.356	0.304	0.384

MAE and MSE were computed for the numerical attributes and accuracy for the categorical attributes

There is no significant difference of single feature predictive capabilities between models, with the overall error proportion being the same throughout the features (tables 4.1 and 4.2).

Lastly, analysing the overall performance of the autoencoders (table 4.3), it can be concluded that the imputed values are hardly any better than the simple baseline solution. Meaning, the autoencoders did not achieve enough reconstruction capacity and, therefore, the observations with missing values should be addressed by removal.

After deletion, the number of observations is now: **379587 ->138794**.

Chapter 5

Modelling

This chapter revolves around the experimental stage: the testing and building of predictive models with the most fitting parameters, the assessment of performance on unseen data and the finishing results of the experiments.

5.1 Data Preprocessing

Each predictive task has its own set of predictors. For every target, the remaining targets are dropped (e.g. for target *air time* - *taxi in*, *taxi out* and *actual block time* are removed). For *taxi out*, the destination weather data is also dropped while for *taxi in* the origin weather data is dropped. Finally, the number of predictors for each predictive task are as follows:

- *air time:* **36 -> 32**
- *taxi out:* **36 -> 26**
- *taxi in:* **36 -> 26**
- *actual block time:* **36 -> 32**

The data was split into a 80-20 train-test set for roughly 100000 instances to train with and 20000 observations of unseen data.

5.1.1 Training pipeline

The training pipeline was comprised of:

- a **categorical encoder**, namely, Leave-One-Out because it does not bring ordinal-based derived correlation between features (such as Label encoding) and does not increase the dimensionality of the training set.

- an **algorithm for dimensionality reduction**: PCA for data compression while keeping a fairly accurate representation of the original data; χ^2 (chi-squared) for feature selection to remove features independent of the target.
- a **scaler**, dependant on the feature selection algorithm. For PCA using standard or robust scaling, for χ^2 using normalisation - requires non-negative features.
- a **predictive algorithm** to test.

5.2 Hyperparameter tuning

Hyperparameters are model parameters whose values are set before the learning process begins. They cannot be implicitly learnt by the machine learning algorithms and are used to control and tweak the learning process.

For this task, a grid search algorithm was used. This technique consists of an exhaustive search through a manually specified subset of hyperparameters for a learning algorithm. The grid search algorithm assesses the performance of each fit of the selected algorithm using MSE, measured using a 5-fold CV on the training set.

5.2.1 Hyperparameter space

For prediction purposes, linear models can sometimes outperform more complex models, especially in situations with small numbers of training cases. Also, these algorithms can be useful to the ensemble of strong predictors. The tuning of Ridge, Lasso and Elastic Net is mainly done tweaking the penalty values. The following sets were tested inside the grid search algorithm:

- For **Ridge regression**, the **L2 regularisation** value was chosen from the set [0.01, 0.05, 0.1, 0.2, 0.5, 1, 2].
- For **Lasso**, the **L1 regularisation** value was chosen from the set [1e$-$6, 5e$-$6, 1e$-$5, 5e$-$5, 1e$-$4, 5e$-$4].
- For **Elastic Net**, the **alpha** (constant that multiplies the penalty values) was chosen from [1e$-$6, 1e$-$5, 1e$-$4] and the **L1 ratio** (mixing parameter, L1 ratio = 0 is an L2 penalty, and L1 ratio = 1 is an L1 penalty) was chosen from the set [0.7, 0.8, 0.9, 0.99].

Gradient Boosting, XGBoost and Random Forest are the most popular ensemble algorithms for regression problems, and the most effective at it, as well. Their performance is substantially based on the number of estimators and learning rate.

- For **XGBoost**, the **L2 regularisation** value chosen from the set [5e$-$1, 1e$-$4, 1e$-$3, 1e$-$2, 0.1], the **L1 regularisation** value was chosen from the set [1e$-$3, 1e$-$2, 0.1, 1] and the **learning rate** value was chosen from the set [0.01, 0.05, 0.1, 0.3].

- For **Gradient Boosting**, the **learning rate** was chosen from the set [1e−2, 5e−2, 1e−2, 0.1].

With the increasingly more powerful machines and data availability, neural networks are also very popular now. For the **Feed-Forward Neural Network**, the tested hyperparameters were: the **depth of the network**, from 1 to 3 hidden layers with multiple layer dimensions; **dropout** from the set [0, 0.2, 0.5]; **batch size** from the set [128, 256, 512].

The dimensionality reduction techniques require a predefined number of features to use. This number is also tested inside the grid search algorithm, ranging from 27 (previously explored best number of principal components (figure 4.3)) to 32 (total number of features), for *air time*. This testing proportion is maintained for the other targets' sets of features.

5.3 Results

5.3.1 Best hyperparameters

The following were retrieved from grid search testing with cross-validation on the *air time* feature set and were the hyperparameters that led to better performance. The testing sets were chosen from experience or existing good performing setups from the machine learning community:

- **Ridge**: L2 - 0.5
- **Lasso**: L1 - 1e−6
- **Elastic Net**: alpha - 1e−4; l1 ratio - 0.9
- **XGBoost**: L2 - 0.1, L1 - 1e−3; learning rate - 0.1
- **Gradient Boosting**: learning rate - 0.05
- **Neural Network**: 64 − 128 − 256 − 128 − 64, dropout - 0.2; batch size - 256

Ridge performed better with the whole set of features while the other models improved with the use of PCA, the best solutions ranging from 27 to 31 principal components. All the estimators got better performance with robust scaling rather than standard scaling.

The tree-based algorithms were trained with 500 estimators.

5.3.2 Training results for best parameters

A naive baseline model using the mean *air time* value results in an MAE of 85, which is far exceed by the evaluations obtained in table 5.1.

The best performer was a stacking of the linear regressors tested, achieving a slightly better result than the models on their own. Unexpectedly, more sophisticated algorithms as the ensemble ones or NN performed significantly worse than the linear models. Worth noting that the best L1 ratio for Elastic Net was of 0.9, converging for a similar algorithm to Lasso, which explains their similar evaluation.

Table 5.1: Comparison of cross-validated performances per algorithm using best performing hyperparameters (using the *air time* target).

algorithms	MAE		R^2		RMSE	
	mean	std	mean	std	mean	std
Ridge	7.795	0.043	99.267	0.028	11.010	5.283
Lasso	7.791	0.043	99.267	0.029	11.011	5.327
Elastic Net	7.791	0.043	99.267	0.029	11.011	5.326
XGBoost	41.676	1.461	81.713	1.797	55.010	17.549
Gradient Boosting	24.445	1.154	94.299	0.517	30.708	9.317
Random Forest	48.741	0.124	72.558	0.510	67.350	76.357
Model Stacking*	7.610	0.0352	99.293	0.031	10.817	2.368
Feed-Forward Neural Network	50.041	8.264	52.288	9.628	89.174	10.197

* first level built from Ridge, Lasso and Elastic Net and meta-estimator as a Gradient Boosting (using best tuned parameters)

5.3.3 Predictions on unobserved data

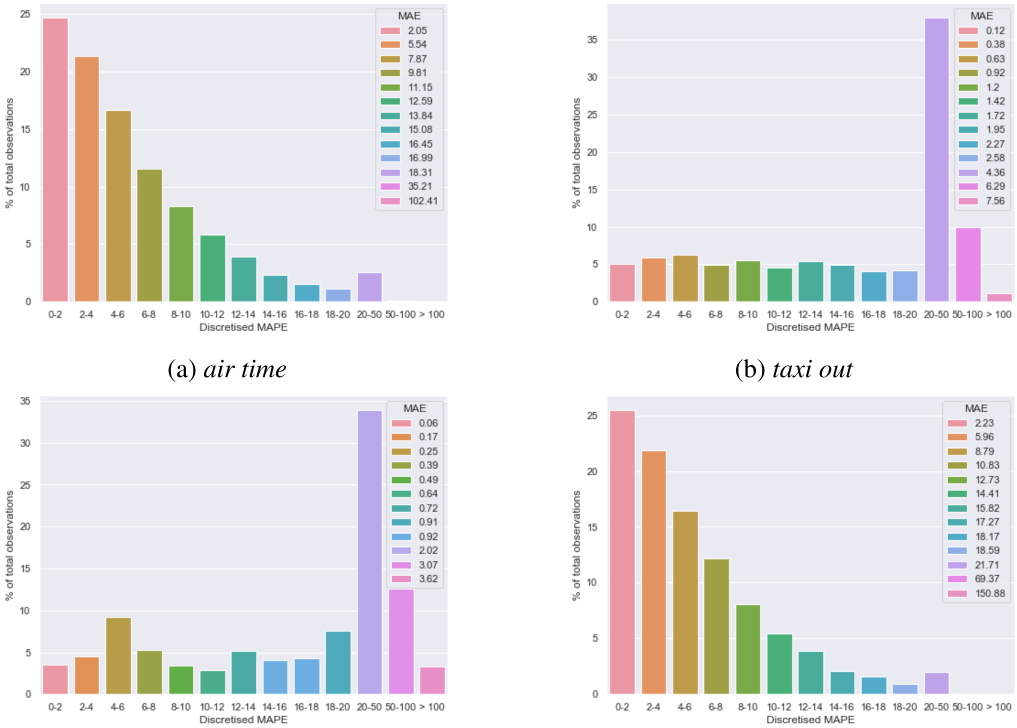
The assessment of algorithm performance above was done using *air time* as the target, and the best setup served as the basis for the remaining target predictions. The dimensionality reduction was tuned individually for each target. As seen in table 5.2, the tuned setup predicts *block time* as well as *air time* while not so well for the *taxis*. After this assessment, some tuning was done (not so thorough as for *air time*) and yet stacking was still the best performer among the various algorithms tested.

Table 5.2: Performance of model stacking in test set for every predictive task. Added row with mean values per target for metric scale comparison.

	air time	*taxi-in*	*taxi-out*	*block time*
mean	162.88	5.59	12.95	181.42
MAE	7.657	1.467	3.002	8.349
RMSE	10.758	2.290	4.100	11.481
MAPE	5.996	29.090	25.272	5.665
R^2	99.309	35.816	25.797	99.232

Through the analysis of figure 5.1, for the *air time*, roughly 25% of prediction can be expected to have a relative error of about 2%, which accounts for an MAE of 2.6 minutes. Further, about 73.% of predictions are expected to have an MAE of under 10 minutes. Roughly the same applies to the *block time* prediction.

For the *taxi times*, while low MAE and RMSE may suggest good performance, the metrics of relative error suggest otherwise. For a mean values of 5.59 for *taxi-in* and of 12.95 for *taxi-*



(a) *air time*

(b) *taxi out*

(c) *taxi in*

(d) *actual block time*

Figure 5.1: MAPE discretisation for each predictive task on stacking's prediction of test set.

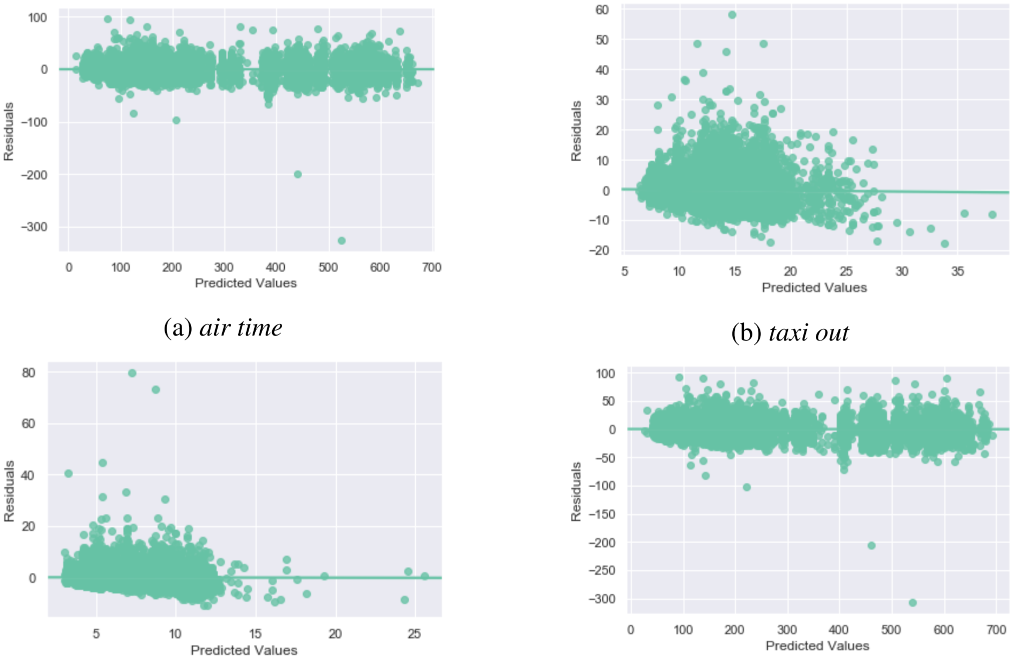
out, the MAE and RMSE are actually high. Furthermore, in figure 5.1, the errors are almost all within the [20-50] bin. This problem should be addressed with further testing of algorithms and hyperparameter tuning.

From the analysis of figure 5.2, some significant outliers can be seen. These instances are generally coming from long-haul flights (e.g. the two most significant residuals from plot 5.2a are from WB instances), which are a minority class in this dataset and significantly skew the error and target distribution. This issue can be addressed by developing different models for wide and narrow-body flights.

Fleet split results

Splitting the dataset into NB and WB subsets results in shapes of 99089 and 11520 respectively. This approach was made just for *air time* and *block time* predictions since it does not make much sense for the size class of an aircraft to affect its taxi times.

As seen in table 5.3, there is a substantial improvement in modelling the WB instances alone. For WB instances, that have a mean value of 500 minutes of travel, there is an MAE of around 13 that results in a MAPE of 2.7% when compared to a MAPE of 6% for the whole dataset. The modelling of NB alone did not bring significant improvement, mainly because this subset comprises almost 90% of the whole dataset (as seen in figure 3.7). These NB results emphasise



(a) *air time*

(b) *taxi out*

(c) *taxi in*

(d) *actual block time*

Figure 5.2: Residuals plot for stacking's prediction on unseen data.

Table 5.3: Comparison of performance dependent on dataset. Added row with mean values per dataset for metric scale comparison.

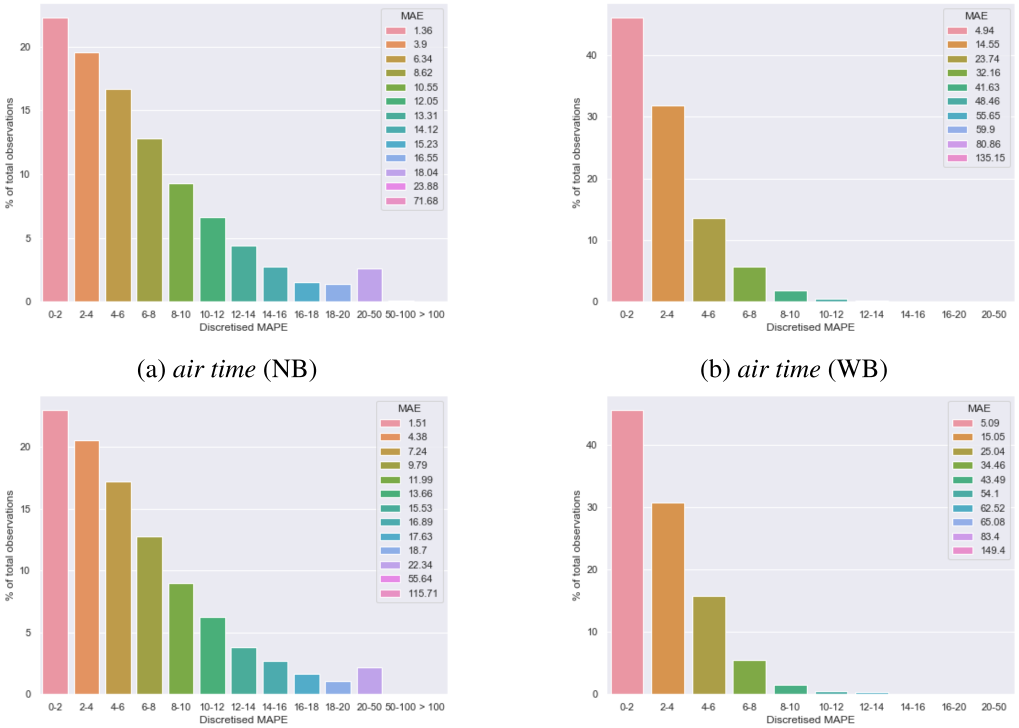
metrics	*air time*			*block time*		
	whole	NB	WB	whole	NB	WB
mean	162.89	122.99	499.64	181.42	141.07	522.11
MAE	7.657	6.953	13.214	8.349	7.694	13.983
RMSE	10.758	9.442	17.027	11.481	10.347	18.062
MAPE	5.996	6.397	2.695	5.665	6.012	2.717
R^2	99.309	97.106	94.708	99.232	96.629	94.149

the hypothesis of the WB instances being misclassified when joint with NB instances since the predictive scores on the whole dataset are the same as just on the NB instances.

The following conclusion can be made from the analysis of figure 5.3:

- For the assessment of the ***air time* in a WB flight**, it is expected for 64.9% of predictions to be under 15 minutes of deviation of the actual air time (threshold for flight delay), and **92.8% to be under 30 minutes of deviation**. For the assessment of the ***block time* in a WB flight**, it is expected for **62.5% of predictions to be under 15 minutes of deviation** of the actual block time, and **90.9% to be under 30 minutes of deviation**.

- For the assessment of the ***air time* in a NB flight**, it is expected for **90.1% of predictions to be under 15 minutes of deviation** of the actual air time, and **99.3% to be under 30**



(a) *air time* (NB)

(b) *air time* (WB)

(c) *block time* (NB)

(d) *block time* (WB)

Figure 5.3: MAPE discretisation for stacking's prediction of test set with dataset's fleet subsets.

minutes of deviation. For the assessment of the ***block time* in a NB flight**, it is expected for **87.7% of predictions to be under 15 minutes of deviation** of the actual block time, and **98.9% to be under 30 minutes of deviation**.

- For the assessment of the ***taxi in* in a flight**, it is expected for **77.8% of predictions to be under 2 minutes of deviation** of the actual taxi-in, and **96.9% to be under 5 minutes of deviation**.

- For the assessment of the ***taxi out* in a flight**, it is expected for **43% of predictions to be under 2 minutes of deviation** of the actual taxi-out, and **83.7% to be under 5 minutes of deviation**. In this work, the *taxi out* prediction did not reach the benchmark set in the work of Balakrishna *et al.* of 81% of predictions being within 2 minutes of deviation [BGS10].

Feature importance

The following feature importance was computed using Permutation Importance. This algorithm allows the measurement of feature importance after model fitting by looking at the changes in evaluation metrics when a feature is not available. The feature importance is computed as the difference between the firstly fitted model and the performance on the permuted test set (test set without a previously selected feature).

The most important features for each predictive task were the following (coefficients expressed in percentage and values under 1% were dismissed.):

- ***air time***:
  - whole/NB: *scheduled block time*: 0.992
  - WB: *destination airport*: 0.509; *origin airport*: 0.398; *scheduled block time*: 0.076
- ***taxi in***: *destination airport*: 0.879; *tail number*: 0.046; *aircraft model*: 0.027
- ***taxi out***: *origin airport*: 0.61; *scheduled departure date month*: 0.064; *origin visibility*: 0.046; *scheduled arrival date month*: 0.045; *tail number*: 0.03; *fleet*: 0.028; *origin air temperature*: 0.023; *scheduled departure date hour*: 0.02; *aircraft model*: 0.019; *prev delay code*: 0.018; *destination airport*: 0.016; *origin cloud coverage*: 0.016; *is night*: 0.012; *scheduled departure date minute*: 0.011; *scheduled block time*: 0.01
- ***block time***:
  - whole/NB: *scheduled block time*: 0.993
  - WB: *destination airport*: 0.501; *origin airport*: 0.399; *scheduled block time*: 0.076

5.3.4 Result assessment

Contrarily to remarks made in Coy's work [Coy06], for this work, weather features were not significant predictors. Also noted on the same work, icing accretion significantly impacts predictions and intuitively other forms of severe weather conditions like strong winds or high precipitation as well. These conditions were absent, which may be the intuition for the low impact of weather features in the predictions.

Without *en-route* weather information, weather data does not retain information from anything but from take-off and landing. Furthermore, the *taxis* are times from operations on an airport, which makes their prediction even more dependent on the weather state of that airport. Without instances with extreme weather conditions (which lack in the dataset due to missing-value trimming and primary instance representation being from domestic operations), the weather data becomes irrelevant for the predictive tasks. This idea may be the intuition behind the poor fit on *taxis* predictions.

Through analysis of the feature importance, it is noticeable that the *taxis* predictions are very correlated with the airport in question. This relationship is due to the model's mapping of the implicit structures that the airport feature translates (like, airports crowdedness or capacity). The presence of these features would positively impact the reliability of the predictions.

The lack of data in the original dataset and consequently severe trim in its size due to missing weather data might have led to the lack of enough data for those more sophisticated algorithms to perform well, such as the ensemble algorithms or neural networks. Neither was Random Forest nor Gradient Boosting techniques [RB14] the best estimators for this dataset.

The prediction of *air time* and *taxi-in* seems to be new work on the literature. The prediction of *block time* seems to have exceeded the other literature work addressing the same issue [Coy06]. The prediction of *taxi-out* was not better than work already developed, namely [BGS10].

Chapter 6

Conclusions and Future Work

This final chapter presents a brief overview of all the developed work, what was achieved, and points out possible directions for future work.

6.1 Conclusions

With the volume of air transport demand increasing more and more, there is also growth in aviation traffic and overcrowded airports. More investment is required in management in the aviation industry, namely in delay assessment. Accurate movement predictions can indirectly contribute to a better disruption recovery process and overall airline management. In turn, reduce the costs significantly in flight delays which may also cover other operational areas, like reducing airport congestion and improving customer satisfaction.

In this work, the flight movement is predicted using flight on-time performance and weather data. Several missing values were present in the dataset, and an attempt on imputation using autoencoders was developed, but unsuccessfully. The final dataset shape was of roughly 200 thousand records.

The built models for the flight time and block time achieved high performance. The taxi-out and taxi-in models need improvement but still achieved satisfactory results. When embedded in a decision support system (namely MASDIMA), should end up contributing for an overall better assessment of disruption impact and inherent mitigation of delay propagation.

6.2 Future Work

This work focuses on predictions made up to 24 hours in advance and never after the scheduled departure, which could be expanded to accommodate use-cases where the aircraft is already airborne.

A cost-sensitive approach would also be interesting to test, since, in these kinds of problems, under-predicting can lead to significant disruptions while over-predicting buffers the flight on-time performance.

A more thorough analysis of the *taxis* prediction would also be useful.

References

[AL15] H. Alonso and A. Loureiro. Predicting flight departure delay at porto airport: A preliminary study. In *2015 7th International Joint Conference on Computational Intelligence (IJCCI)*, volume 3, pages 93–98, Nov. 2015.

[BBD$^+$10] Michael Ball, Cynthia Barnhart, Martin Dresner, Mark Hansen, Kevin Neels, Amedeo Odoni, Everett Peterson, Lance Sherry, Antonio Trani, Bo Zou, Rodrigo Britto, Doug Fearing, Prem Swaroop, Nitish Uman, Vikrant Vaze, and Augusto Voltes. Total delay impact study: A comprehensive assessment of the costs and impacts of flight delay in the united states. Technical report, NEXTOR II, Nov. 2010.

[BGS10] Poornima Balakrishna, Rajesh Ganesan, and Lance Sherry. Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of tampa bay departures. *Transportation Research Part C: Emerging Technologies, Special issue on Transportation Simulation Advances in Air Transportation Research*, 18(6):950 – 962, 2010.

[Bre96] Leo Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, July 1996.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct. 2001.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[CKBM16] S. Choi, Y. J. Kim, S. Briceno, and D. Mavris. Prediction of weather-induced airline delays based on machine learning algorithms. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–6, Sept. 2016.

[CKBM17] S. Choi, Y. J. Kim, S. Briceno, and D. Mavris. Cost-sensitive prediction of airline delays using machine learning. In *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, pages 1–8, Sept. 2017.

[CKD$^+$19] Navoneel Chakrabarty, Tuhin Kundu, Sudipta Dandapat, Apurba Sarkar, and Dipak Kumar Kole. Flight arrival delay prediction using gradient boosting classifier. In Ajith Abraham, Paramartha Dutta, Jyotsna Kumar Mandal, Abhishek Bhattacharya, and Soumi Dutta, editors, *Emerging Technologies in Data Mining and Information Security*, pages 651–659, Singapore, 2019. Springer Singapore.

[CLLR10] Jens Clausen, Allan Larsen, Jesper Larsen, and Natalia J. Rezanova. Disruption management in the airline industry—concepts, models and methods. *Computers & Operations Research*, 37(5):809 – 821, 2010.

[Coy06] Steven Coy. A global model for estimating the block time of commercial passenger aircraft. *Journal of Air Transport Management*, 12(6):300 – 305, 2006.

[CRO14] António J. M. Castro, Ana Rocha, and Eugénio Oliveira. *A New Approach for Disruption Management in Airline Operations Control*, volume 1. Springer, Berlin, Heidelberg, Sept. 2014.

[CT15] A. Cook and G. Tanner. European airline delay cost reference values. https://www.eurocontrol.int/sites/default/files/publication/files/european-airline-delay-cost-reference-values-final-report-4-1.pdf, 2015. Visited on 2020-02-02.

[DA12] Vinayak Deshpande and Mazhar Arıkan. The impact of airline flight schedules on flight delays. *Manufacturing & Service Operations Management*, 14(3):423–440, 2012.

[dC13] António Jesus Monteiro de Castro. *A Distributed Approach to Integrated and Dynamic Disruption Management in Airline Operations Control*. Phd thesis, Faculty of Engineering, University of Porto, Portugal, Porto, Portugal, July 2013.

[Die00] Thomas G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, page 1–15, Berlin, Heidelberg, 2000. Springer-Verlag.

[DYC+20] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, Apr. 2020.

[ESB17] Reza Etemad-Sajadi and Laura Bohrer. The impact of service recovery output/process on customer satisfaction and loyalty: The case of the airline industry. *Tourism and Hospitality Research*, 19:146735841774308, Dec. 2017.

[EUR17] EUROCONTROL. Coda digest: All-causes delay and cancellations to air transport in europe - 2017. https://www.eurocontrol.int/sites/default/files/publication/files/coda-digest-annual-2017.pdf, 2017. Visited on 2020-02-02.

[EUR18] EUROCONTROL. Coda digest: All-causes delay and cancellations to air transport in europe - 2018. https://www.eurocontrol.int/sites/default/files/2019-05/coda-digest-annual-2018.pdf, 2018. Visited on 2020-02-02.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[HH14] Lu Hao and Mark Hansen. Block time reliability and scheduled block time setting. *Transportation Research Part B: Methodological*, 69:98 – 111, 2014.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning. *Aug, Springer*, 1, Jan. 2001.

[HZL+13] Z. Huang, G. Zweig, M. Levit, B. Dumoulin, B. Oguz, and S. Chang. Accelerating recurrent neural network training via two stage classes and parallelization. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 326–331, Dec. 2013.

REFERENCES

[IAT18] IATA. Industry statistics - fact sheet. https://www.iata.org/pressroom/facts_figures/fact_sheets/Documents/fact-sheet-industry-facts.pdf, 2018. Visited on 2019-10-21.

[KCBM16] Y. J. Kim, S. Choi, S. Briceno, and D. Mavris. A deep learning approach to flight delay prediction. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–6, Sept. 2016.

[KTK16] Sina Khanmohammadi, Salih Tutun, and Yunus Kucuk. A new multilevel input layer artificial neural network for predicting flight delays at jfk airport. *Procedia Computer Science*, 95:237 – 244, 2016.

[KW19] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *CoRR*, abs/1906.02691, 2019.

[MBK+17] S. Manna, S. Biswas, R. Kundu, S. Rakshit, P. Gupta, and S. Barman. A statistical approach to predict flight delay using gradient boosted decision tree. In *2017 International Conference on Computational Intelligence in Data Science(ICCIDS)*, pages 1–5, June 2017.

[MKA18] John T. McCoy, Steve Kroon, and Lidia Auret. Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine*, 51(21):141 – 146, 2018.

[RB14] Juan Jose Rebollo and Hamsa Balakrishnan. Characterization and prediction of air traffic delays. *Transportation Research Part C: Emerging Technologies*, 44:231 – 241, 2014.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[TSS+17] B. Thiagarajan, L. Srinivasan, A. V. Sharma, D. Sreekanthan, and V. Vijayaraghavan. A machine learning approach for prediction of on-time performance of flights. In *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, pages 1–6, Sept. 2017.

[Uni] Iowa State University. Iowa environmental mesonet. https://mesonet.agron.iastate.edu/request/download.phtml. Visited on 2019-07-30.

[VLBM08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, Helsinki, Finland, 2008.

[WZHC19] Yanjun Wang, Ying Zhou, Mark Hansen, and Christopher Chin. Scheduled block time setting and on-time performance of u.s. and chinese airlines—a comparative analysis. *Transportation Research Part A: Policy and Practice*, 130:825 – 843, 2019.

[YGA+19] Bin Yu, Zhen Guo, Sobhan Asian, Huaizhu Wang, and Gang Chen. Flight delay prediction for commercial air transport: A deep learning approach. *Transportation Research Part E: Logistics and Transportation Review*, 125:203 – 221, 2019.