PREDICTING COMMERCIAL U.S. FLIGHT DELAYS USING MACHINE LEARNING MODELS AND TABNET

ROB VAN LOENHOUT

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

STUDENT NUMBER

2088610

COMMITTEE

Dr. Drew Hendrickson
Dr. Murat Kirtay

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

December 4th, 2023

WORD COUNT

8713

ACKNOWLEDGMENTS

I wish to extend my sincere appreciation to my parents for consistently facilitating and supporting my continuous pursuit of knowledge, as well as for encouraging me to confront challenges at each stage of my academic journey. Additionally, I express gratitude to my friends who contributed to maintaining my mental well-being during the demanding phases of thesis work, offering welcome distractions when needed. Lastly, I wish to convey my deepest thanks to my supervisor, Dr. Drew Hendrickson, for his invaluable feedback and the weekly meetings that consistently motivated and guided me throughout the thesis-writing process.

PREDICTING COMMERCIAL U.S. FLIGHT DELAYS USING MACHINE LEARNING MODELS AND TABNET

ROB VAN LOENHOUT

Abstract

Air travel stands out as one of the most widely embraced means of transportation between continents and major cities. Nevertheless, delays significantly impact travelers' experiences, incur costs for airlines, and contribute to environmental emissions. This thesis endeavours to mitigate these delays through predictive modelling, utilizing a combination of U.S. flight, aircraft (BTS), weather (NOAA), and demographic data (USCB). The central research question guiding this study is: *To what extent can U.S. flight delays be predicted on imbalanced data using Machine/Deep learning models, and how do airlines differ in terms of the predictability of delays?* The problem is framed as a binary classification task, focusing on predicting whether a delay has occurred.

In contrast to existing literature, a noteworthy aspect of this thesis is the application of TabNet to the problem. The objective is to assess whether a more complex model can outperform established models. Additionally, recognizing the imbalanced nature of the data, three resampling techniques (over-, under-, and hybrid resampling) are explored to evaluate their impact on performance. The study identifies XGBoost, employing a hybrid resampling technique, as the most accurate solution for the problem, achieving an AUROC score of 0.819. Although TabNet's performance fell below expectations, an engineered feature examining whether a previous flight was delayed emerged as a significant predictor of delays. Lastly, a distinct variation in prediction errors among airlines was observed, with Hawaiian Airlines proving the least predictable and Frontier Airlines the most predictable.

1 DATA SOURCE/ETHICS/CODE/TECHNOLOGY (DSECT) STATEMENT

Work on this thesis did not involve collecting data from human participants or animals. The original owner of the data and code used in this thesis retains ownership of the data during and after completion of this thesis. Figures used in this thesis have either been created or used under the CC BY 4.0 license (Figure 7). ChatGPT-3.5 was used for code debugging and spelling checks. Moreover, Grammarly was used as an additional tool to improve sentence structures and grammar. The code used in this thesis is publicly available on GitHub using the following link. A more detailed overview of the packages, software and hardware used can be found in Appendix B.

2 INTRODUCTION

Air travel is a prevalent mode of long-distance transportation, appreciated for its convenience and speed, especially for intercontinental and major city journeys. In 2022, over 853 million passengers travelled by air in the United States, marking a 30% increase compared to 2021 (Bureau of Transportation Statistics, 2023b). This figure is rapidly approaching pre-COVID travel numbers, which stood at 927 million passengers in 2019.

As the number of passengers continues to rise, so do flight delays. In 2022, flight delays in the United States increased by 1.36% compared to pre-COVID 2019 figures (Bureau of Transportation Statistics, 2023d). These delays impact not only airlines but also the passengers traveling with them. Airlines incur monetary costs, such as fines paid to passengers and subsequent delays caused by initial disruptions, along with aircraft downtime. An estimated $101.18 per minute of operating costs is lost due to delays (Airlines For America, 2023). Additionally, there is the risk of reputational damage, potentially affecting their income. Passengers may not incur direct monetary costs, but delays can significantly affect their travel experience. Missed layovers, overnight stays due to long delays, and the potential loss of time from work or other obligations all contribute to this impact. A study by Stone (2018) revealed that one in six travel itineraries with connecting flights from small airports to larger hubs would be disrupted due to delays, causing subsequent longer stays at the larger hubs or even overnight stays. Notably, within U.S. airlines, differences in delay rates exist, with some airlines more prone to delays than others. For example, JetBlue Airways had an increase of 5.2% in delays in 2022 compared to 2019, while Delta only had an increase of 1.36%, and United Airlines actually had a decrease in delays by 2.66% (Buchholz, 2023).

This thesis aims to predict these delays by leveraging flight delay data from the Bureau of Transportation Statistics (BTS) in conjunction with relevant features from datasets like the National Oceanic and Atmospheric Administration (NOAA) weather data and aircraft inventory information by the BTS to improve predictability and thus lower the chance of unexpected delays. Furthermore, a disparate group analysis will be made on the predictability per airline because, as shown, this can differ drastically.

2.1 *Societal and Scientific Relevance*

As mentioned, air travel is a widely used means of transportation for long distances, often proving more cost-effective than alternatives like public transport or driving. Therefore, flight delays unquestionably affect air travellers and society at large. From an environmental standpoint, these delays can lead to unnecessary environmental impacts. It is estimated that delays cause about 1% to 1.5% more fuel consumption, which could be mitigated through the use of delay predictions (Ryerson et al., 2014).

From a scientific perspective, it is noteworthy that many papers identified in the literature review did not make their code publicly available. To enhance transparency and contribute to the scientific community, I intend to make my code accessible for others to use and build upon while maintaining reproducibility. Furthermore, in the formulation of the research questions, the application of the TabNet method to this problem type has not been explored. Through this research, I aim to establish a baseline using TabNet for further studies in this area.

2.2 *Research Questions*

Based on earlier stated problems the following main research question has been formulated:

> *To what extent can U.S. flight delays be predicted on imbalanced data using Machine/Deep learning models and how do airlines differ in terms of predictability of delays?*

The main research question is divided into three sub-questions:

RQ1 *Among XGBoost, Logistic Regression, Random Forests and TabNet, Which models and hyperparameters perform the best when classifying imbalanced delays?*

These models have been selected to compare with TabNet based on the literature review in Section 3.1. These decisions and hyperparameter choices will be further elaborated on in Chapter 4.

RQ2 *Do class-imbalance strategies like over-, under-, and hybrid-sampling improve the models' predictive capabilities?*

As mentioned earlier, an increase in delays can be noted. With around 19.95% of flights being delayed for 2019, this means that the other 80.05% of flight data is on-time or cancelled, indicating that a class-imbalance might be present (Bureau of Transportation Statistics, 2023d). Because of this, a research question has been formulated on how to solve this imbalance and if resampling actually improves the selected models. A more detailed explanation will be given in Chapter 4.

RQ3 *How do airlines compare in terms of prediction errors made by the best model and how are they distributed?*

As noted before, there is a clear difference between airlines and their on-time performance. Therefore, it would be interesting to analyse how well the model can predict delays per airline and if the model makes more mistakes for a given airline than others. This will be done by analysing the prediction errors made by the best model from RQ1 or RQ2 and grouping these results per airline.

3 LITERATURE REVIEW

3.1 *Models Used in Related Studies*

The literature is assessed to make an informed decision on the models used to compare with TabNet. Here, the most promising models will be chosen, forming a realistic comparison. The eXtreme Gradient Boosting (XGBoost) algorithm stands out as a frequently used model within the flight delay prediction domain. A paper by Chakrabarty et al. (2019) utilized the model, comparing it to k-NN, SVM, and RF. XGBoost achieved the highest accuracy and an AUROC score of 0.54 using U.S. BTS data. In the same year, Chakrabarty (2019) published another paper utilizing XGBoost, evaluating different pre-processing steps on the BTS data, resulting in an F1-Score of 85.73%. Hatıpoğlu et al. (2022) used data from a Turkish airline company to achieve an impressive AUROC score of 0.958, comparing it to other high-performing models like CatBoost and LightGBM.

Random Forests (RF) were also utilized or compared in various papers. Rahul et al. (2022) applied RF solely to data from airports in New York City, achieving an accuracy of 86%. A paper by Choi et al. (2016) compared RF to AdaBoost, k-NN, and Decision Trees (DT) using BTS data and weather-related data from NOAA, with RF coming out on top with an AUROC score of 0.68. Thiagarajan et al. (2017) not only predicted departure delays but also arrival delays for U.S. data, using classification and regression

models. They compared RF, MLP, Gradient Boosting, AdaBoost, and Extra-Trees, with RF being the most accurate model, obtaining an AUROC score of 0.92 for departure delay prediction and 0.98 for arrival delay prediction. Lastly, J. Chen and Li (2019) used a multi-label RF classifier on BTS and NOAA data for a single airport, achieving an accuracy of 87%.

Some research on deep learning has also been done. A Long Short-Term Memory (LSTM) network was applied by Boggavarapu et al. (2019). They used this model on a single U.S. airport and achieved an R-Squared score of 94.82, accurately estimating the average departure delay and confirming that time-series analysis is also possible. Jiang et al. (2020) also applied a deep learning model in the form of Multi-Layer Perceptron (MLP) and compared it to SVM, DT, and RF. They found that MLPs beat the tree-like methods, proving that applying more complex models could result in higher accuracy. The final result was an accuracy score of 89.07% on their BTS data.

Lastly, some papers utilized white-box type models with equally good results as the more complex models, emphasizing that complex models are not always 'better' than easier-to-understand models. Daldır et al. (2021) showed that a simple model like Logistic Regression can still outperform models like XGBoost, RF, CatBoost, Artificial Neural Networks (ANN), and SVM. It did this with an AUROC score of 0.984 using data from a Turkish international airport. Wang et al. (2017) also applied a regression model to the domain using Multiple Linear Regression (MLR) and compared it to Naïve Bayes and C4.5. MLR had an F1-Score of 79.1%, outperforming the other white-box style methods. In the paper proposed by Dand et al. (2019), they also used a Naïve Bayes model and compared it to RF, k-NN, and DT, with Naïve Bayes beating the others with an AUROC score of 0.621.

3.2 *Data Used by Other Studies*

Almost all papers found in the literature used U.S. data from the BTS, with most of them using subsetting on their data and filtering on a certain amount of airports or a specific airport (Chakrabarty, 2019; Chakrabarty et al., 2019; Manna et al., 2018; Thiagarajan et al., 2017). Four papers used BTS in combination with weather data from NOAA to improve predictive capabilities (Boggavarapu et al., 2019; J. Chen & Li, 2019; Choi et al., 2016; Jiang et al., 2020). Two papers used data from Turkey with very high predictive capabilities, which included weather-related data in the dataset, emphasizing the impact of weather data (Daldır et al., 2021; Hatıpoğlu et al., 2022). A single paper used data from commercial flights in China, which also included weather-related data (Wang et al., 2017). The data

used by the literature ranged from 2013 to 2018, with most papers focusing on a single or two years. Two papers took a longer range of data. The first paper, by Choi et al. (2016), utilized a range of 10 years with weather data but did filter on 45 of the major airports. The second paper, by Dand et al. (2019), used a range of 5 years and included all airports but did not include weather-related data.

3.3 *Class-Imbalance Techniques*

In the domain of flight delays, the actual data will almost always be imbalanced, with more flights being on time instead of delayed. In the literature that was reviewed, only six papers explicitly mentioned trying to resample to fix the class-imbalance. The oversampling strategy was applied by all of these papers with SMOTE, and some SMOTE-related methods only being applied as a minority oversampling technique (Chakrabarty, 2019; J. Chen & Li, 2019; Hatıpoğlu et al., 2022). Three papers applied a hybrid-method as they combined SMOTE with Tomek's Links or Random Undersampling (Choi et al., 2016; Dand et al., 2019; Thiagarajan et al., 2017). Five of the papers reported improved results, while Hatıpoğlu et al. (2022) reported worse performance with SMOTE.

3.4 *Limitations and Recommendations by Other Studies*

The main consensus from the papers that supplied future research was that the data needed to be expanded (Boggavarapu et al., 2019; Daldır et al., 2021; Manna et al., 2018; Thiagarajan et al., 2017). The reason being that, as stated earlier, most papers focused only on the top five, ten, or forty airports or just a single airport. Moreover, suggestions by Chakrabarty et al. (2019), and Boggavarapu et al. (2019) were made to apply more complex models to the problem to uncover more complex relationships in the data. The only paper that applied the model to all airports available did not yet try to add weather-related data and provided this as a recommendation (Dand et al., 2019).

3.5 *Research Gap*

Firstly, as identified in Section 3.2, there is a clear gap in terms of data being used. Almost all papers found limited their data to a top X or just single airports/routes. Because of this, all available data for 2019 from the BTS will be used with the idea to improve the generalization of the models created. For example, if you only train a model on New York City airports,

it might perform worse in places with, let's say, a different climate. If a general model can be created that predicts delays in general, that would be more sustainable in terms of resources. Moreover, more data could mean better predictions in general as the models are subjected to more information that could have an impact on a delay.

Secondly, some papers suggested using more complex models like deep learning models to find more complex relationships within the data. Because of this, the tabular deep learning model TabNet will be tested and compared to proven models within the domain. It has proven itself to be as accurate or more than some of the high-performing models, more on this in Section 4.4.4.

Thirdly, some papers already applied oversampling and hybrid methods to the imbalanced data. However, the main oversampling method used was SMOTE while others have not been tested. Therefore, it might be beneficial to test other methods like Random Oversampling to see if these improve predictability. Moreover, hybrid methods have also been tried but again only with SMOTE-methods; here, it could also be beneficial to test other methods like Random Oversampling combined with Random Undersampling.

Lastly, we have to identify the State-Of-The-Art (SOTA) from which this thesis will be compared with. Because most papers focus only on single or a limited number of airports, it would not be fair to compare with these papers. The only paper that did not filter on airports was the paper by Dand et al. (2019). They used data from BTS from 2012 till 2017 but did not include weather-related data, which they did state as a future research suggestion. With this, they achieved a 0.62 AUROC score using Naïve bayes; using this, we can see if including weather will improve the predictability for delays using all airports in the BTS dataset for 2019.

4 METHOD

In this chapter an overview of the methods used for this thesis will be shown. Figure 1 is a visual representation of the data science pipeline that was applied. This is a high-level summary of the steps taken to achieve the results.

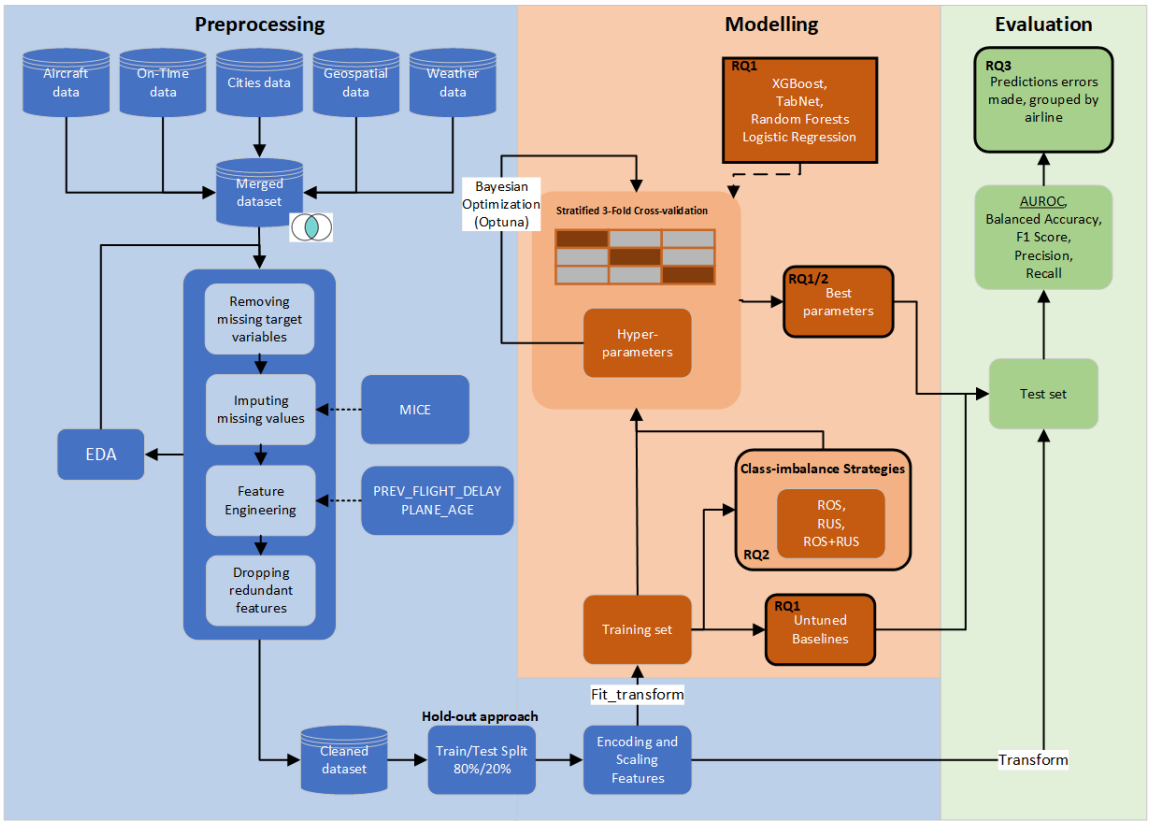

Figure 1: An overview of the Data Science pipeline

4.1 *Dataset description*

The dataset used in this study has been created by merging several different .CSV data files from four sources. Data focusing on delays, scheduling, distances, and aircraft information was gathered from the BTS (Bureau of Transportation Statistics, 2023a, 2023c). Weather data was collected from the NOAA, providing daily weather summaries near airports (National Oceanic and Atmospheric Administration, 2023). Geospatial data, containing longitude and latitude information of airports, was obtained from ip2location (IP2Location, 2023). Demographic data about U.S. cities was gathered from USCB, displaying information about population, median age, and average household size (U.S. Census Bureau, 2023).

It is important to note that not all datasets originated from the same year. The weather, delay, and aircraft data came from 2019, the demographic data from 2015, and the geospatial data from 2023. This could pose certain limitations on the accuracy of the model and contribute to a poor generalization of the model. However, only the demographic data could

pose a problem, as these are the only time-dependent features from a different year.

During merging, some features from the datasets were omitted as they did not contribute or could skew the predictive performance of the model, such as actual departure time, actual arrival time and delay in minutes. Other features like tail number and airport ID were retained but omitted after preprocessing. The dataset, after merging, contains 6,808,079 entries and 25 features. In Appendix A, a more detailed description of each feature can be found.

4.2 *Exploratory Data Analysis (EDA)*

EDA is a crucial step in the data science workflow, used to gain an in-depth understanding of the data through statistical methods and visualizations. In this research, box plots and histograms have been used to understand the distribution and detect possible outliers (see Appendix F). Initially, an assessment of missing data was made. Below, you can find Figure 2 displaying the missing data, where each white bar represents one or more missing entries.

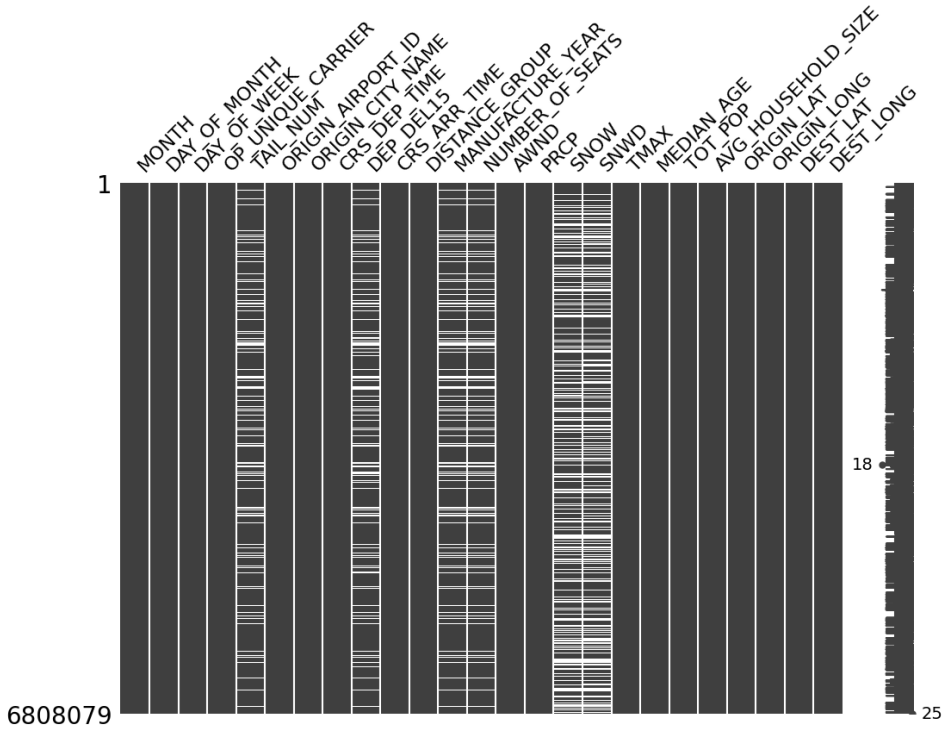

Figure 2: Overview of missing data before dropping values in DEP_DEL15

As shown in Figure 2, the columns: TAIL_NUM, DEP_DEL15, MANUFACTURE_YEAR, N_OF_SEATS, SNOW, and SNWD show a significant number of missing entries. Moreover, AWND, PRCP, and TMAX have a small number of missing entries, not visible on the graph due to the entry count. The other columns do not possess any missing entries.

Next, we explore the distribution of the target variable. In this case, the target variable is a binary classifier, where 0 means there was no delay or it lasted less than 15 minutes, and 1 if the delay exceeded more than 15 minutes. Figure 3 shows this distribution and a clear imbalance within the distribution. Here, 66.34% represents the 0 class, 15.44% the 1 class, and 18.22% are missing.

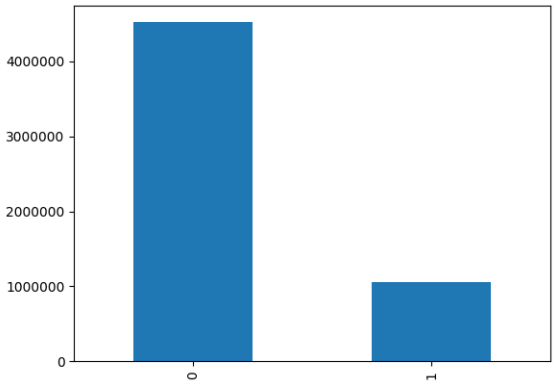

Figure 3: Class imbalance before dropping values in DEP_DEL15

Lastly, to validate that there are no highly correlated features with the target variable, a correlation heatmap was generated after merging and before preprocessing. Figure 4 shows that there are no highly correlated features with the target variable; the ones that stand out are CRS_DEP_TIME and CRS_ARR_TIME, which are the planned departure and arrival times of the plane.

4.3 *Data Preprocessing*

Data preprocessing is a crucial step in the data science pipeline, ensuring that the data is of the utmost quality so that the models can perform accurately. As shown in the previous chapter, there are a number of columns with missing variables. These values can impact the performance of the models and should therefore be imputed with appropriate values or dropped entirely. Furthermore, the data has to be transformed using encoding and scaling techniques before it can be inserted into the models.

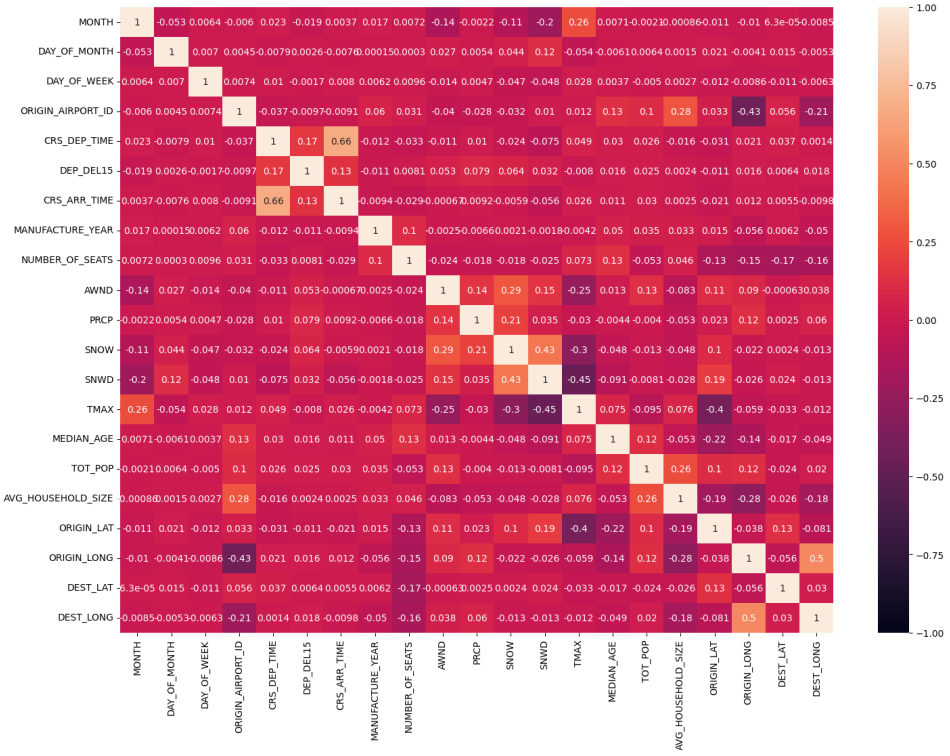

Figure 4: Correlation diagram before preprocessing

4.3.1 *Cleaning missing data*

As shown in the previous paragraph, there are a number of columns with missing variables. These values can impact the performance of the models and should therefore be imputed with appropriate values or dropped entirely.

Firstly, the target variable DEP_DEL15 contained about 1,240,432 missing entries, which is quite a significant number. Upon closer inspection of the original data, it became apparent that these flights were cancelled instead of delayed. Because this study only focuses on delays and does not include predicting cancellations, the missing values were omitted from the final dataset. Consequently, when rerunning an analysis on the missing data, it showed that the missing values for the columns TAIL_NUMBER, MANUFACTURE_YEAR, and NUMBER_OF_SEATS also completely disappeared. This correlation makes sense, as cancelled flights do not have any planes connected to them, and thus no tail number. Moreover, because the manufacturing year and number of seats were merged from a different dataset onto the tail number feature, this would result in the other missing values. The missing values were reevaluated in Figure 5, and after dropping the entries, they summed up to 5,567,978 in total.

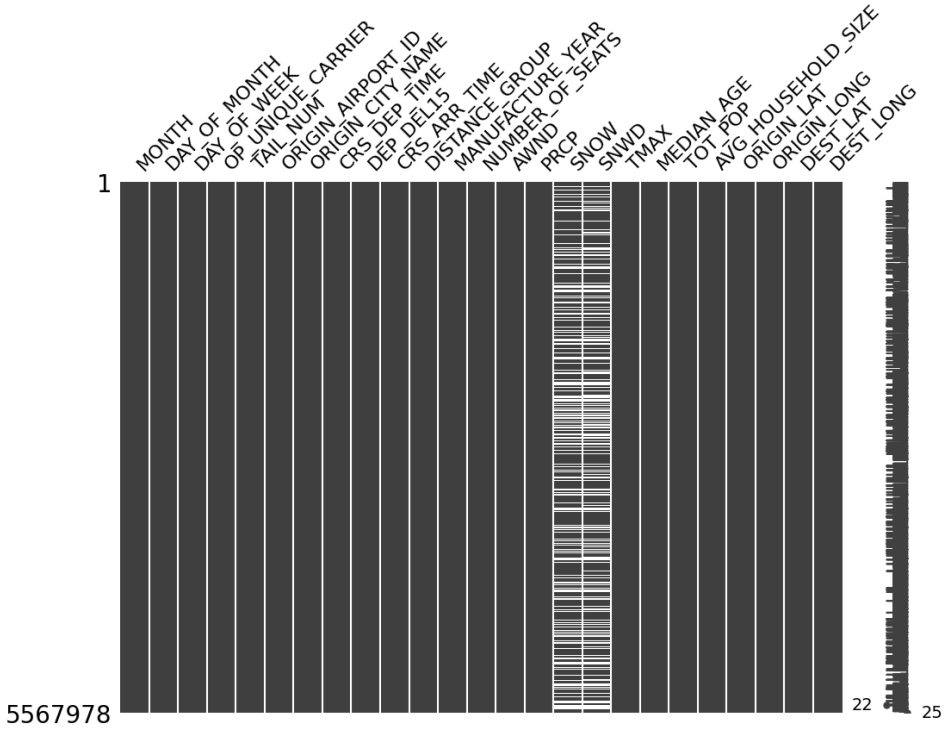

Figure 5: Overview of missing data after dropping values in DEP_DEL15

After re-evaluation, it showed that some weather-related features like SNOW and SNWD still had a high number of missing entries. Moreover, AWND, PRCP, and TMAX still contained a small number of missing entries. In the case of the missing values for SNOW and SWND, an assumption was made. If both values were missing from the columns, it is assumed that the values are both empty, and thus no snow has fallen that day. The reasoning is that these features are correlated, as SNOW implies snowfall in inches, and SNWD implies the depth of snow. When it was found that both instances are missing, they were both imputed with 0.0, instances where one of the two values is missing were left empty. These imputations resulted in a significant drop in missing entries, and a summary can be found in Table 1.

Because the weather features are correlated and can help explain other missing weather entries, a MICE-inspired method was applied to impute the missing entries. When looking back at Figure 4, we can see that some weather features are correlated with each other and can give information about the value that is missing (van Buuren & Groothuis-Oudshoorn, 2011). For this reason, the iterative imputer from the Scikit-Learn package was applied to the five weather-related features. A mean/mode/median imputation was considered but was disregarded as it did not look at other

Table 1: Missing Weather Data Percentage Before and After Imputation

Feature	% Missing Before 0.0 Imputation	% Missing After 0.0 Imputation
SNOW	29.95%	0.17%
SWND	32.38%	2.6%
AWND	0.01%	0.01%
PRCP	0.03%	0.03%
TMAX	0.01%	0.01%

relevant features that could impact the missing/imputed value and could introduce noise instead. Furthermore, it is assumed that the missing data is Missing At Random (MAR), an assumption for the multiple imputation method. MAR transpires when the missingness of the data depends on already existing observations (Pedersen et al., 2017).

4.3.2 *Feature engineering*

To help improve the predictive capabilities of the models, a total of two features were engineered from other features in the dataset. Firstly, a boolean feature has been created that represents if a previous flight from the same plane was also delayed. If it was, this value will be set to 1; otherwise, it will be set to 0. To create this feature, the data had to be ordered by month, day of the month, tail number, and planned departure time. From here, the previous plane can be checked, and if it matches the tail number and was delayed, then the value will be set to 1 to indicate a previous delayed flight. In theory, this would have an impact on the next flight and may cause a delay.

The second feature was created by utilizing the manufacturing year of a plane. Using this feature, the age of the plane was set by subtracting the year of the dataset, which is 2019, from the manufacturing year. The theory is that older planes might need more maintenance and have a tendency to break down more often, thus creating delays.

In addition, the features CRS_DEP_TIME and CRS_ARR_TIME, which stand for the planned departure and arrival times, were edited slightly. These features were formatted in military time format; it could be that in this format, the model might misinterpret these features. To prevent this, these features were converted to use the minutes from midnight instead.

After these changes, the correlation diagram was reevaluated to check the impact of the engineered features. This diagram can be found in Figure 6. What stands out is that the engineered features PLANE_AGE, CRS_DEP_TIME, and CRS_ARR_TIME did not change in their impact on the target variable. Furthermore, the newly engineered PREV_FLIGHT_DELAY

does seem to be somewhat correlated with the target variable. From a domain perspective, this impact can also be explained, and thus this feature will not be further treated with Principal Component Analysis (PCA).

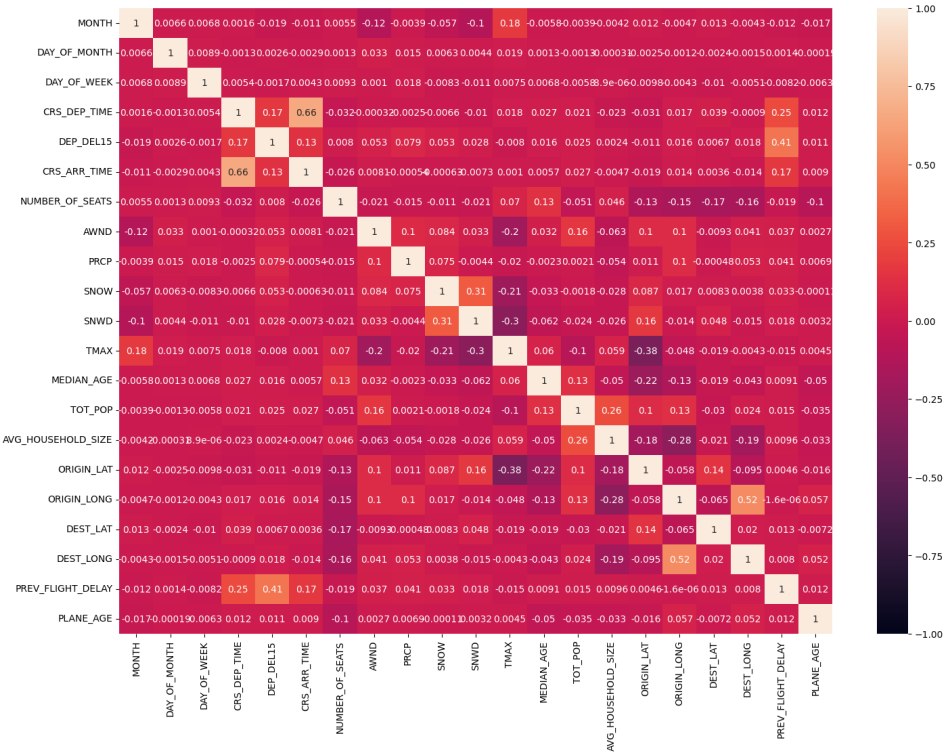

Figure 6: Correlation diagram after preprocessing

4.3.3 *Data splitting*

A stratified hold-out approach will be used to divide the dataset into train and test sets, where training will consist of 80% of the data and testing 20% of the data. This method and numbers are based on previous research in the domain (Chakrabarty, 2019; Dand et al., 2019; Rahul et al., 2022). It should be acknowledged that when using this method, it is possible that the testing set might be divided favourably or unfavourably. However, because of the size of the dataset, a general hold-out approach would suffice in making the testing data representative and generalized (Yadav & Shukla, 2016).

4.3.4 *Dummy encoding and feature scaling*

A total of 22 columns are present in the dataset after feature engineering. Of these features, only OP_UNIQUE_CARRIER is a categorical one, which

gives a two-character abbreviation for an airline. While some models can handle categorical features, others cannot. In this case, Logistic Regression cannot natively handle categorical features. Therefore, we have to encode this feature. At first, a Label Encoder was considered; however, this could introduce an unwanted ordinal relationship and thus misinterpreting the feature (Hancock & Khoshgoftaar, 2020). As a result, dummy encoding was applied. This does increase the dimensionality of the dataset, but since there are only 16 airlines, this would increase the number of columns to 37, which is not a significant increase.

Feature scaling is a widely used practice within the Data Science domain. Scaling ensures that all features use the same scale or unit of measure, thus making it easier to compare the features within a model. Some models are more sensitive to scaled features and thus require it for a fair and optimal evaluation. Moreover, it can help make the models robust to outliers, which is the case for this dataset as it contains some features that suffer from outliers. Logistic regression, TabNet, and XGBoost will benefit from applying a form of feature scaling (Bisong, 2019; Zhen & Zhu, 2023). While Random Forests are quite resilient against the scale of the features and thus will not impact the model significantly. However, because the other models are impacted, a robust scaler from Scikit-Learn will be applied to the data, so to make a fair assessment of all the models and improve convergence.

4.4 *Models and hyperparameters*

In this subsection, we delve into the chosen models for this research: Logistic Regression (LR), Random Forests (RF), XGBoost (XGB), and TabNet. Additionally, we discuss the chosen ranges for the hyperparameters.

4.4.1 *Logistic Regression*

Firstly, logistic regression will be discussed. This model is primarily used for binary classification problems. The advantage lies in its ease of implementation and interpretation as it is a white-box type model. Moreover, if the features in the dataset are easily linearly separable, it will perform accurately without the use of a lot of resources. The downside is that, being a linear model, it cannot identify more complex relationships like non-linear models. However, it provides a nice baseline for other complex models, and if it performs well, it could be argued that more complicated models are too excessive and might be overfitting (Ayer et al., 2010).

Beyond providing a good interpretable baseline model, it also showed the potential of outperforming other complex models in the domain. In a paper by Daldır et al. (2021), they showed that it was their most well-performing model on their Turkish flight delay dataset. For these reasons, the model has been chosen to compare with other more complex models like TabNet and XGB.

The C regularization parameter and its ranges have been based on the research by Daldır et al. (2021). While the solvers have been chosen based on recommendations by Scikit-Learn (Pedregosa et al., 2011). They recommend these solvers for larger datasets, which is the case for this study. For the penalty parameter, not all solvers supported L1-regularization; therefore, the decision was made to only apply L2-regularization or no regularization.

Table 2: Hyperparameter Ranges/Values for Logistic Regression

Hyperparameter	**Ranges/Values**
C regularization	[0.001, 1000], log sampled
Solver	[sag, saga, newton-cg]
Penalty	[L2, None]

4.4.2 *Random Forests*

Random forests are a well-tested decision tree ensemble method known for their robustness against outliers, noisy data, and imbalanced data. Due to the fact that it combines N number of decision trees with random feature selection, it is also less prone to overfitting than, for example, logistic regression. Moreover, it provides insights into the feature importance and, as such, provides a deeper understanding of the more influential features. The downside is that it does require more computational power and training time due to the many Decision Trees it has to create (Breiman, 2001).

The model has been chosen because of its proven performance inside and outside of the delay prediction domain (J. Chen & Li, 2019; Choi et al., 2016; Rahul et al., 2022). Moreover, it has shown to handle large datasets well and accurately due to its design but might take longer to train and needs more computational resources depending on the size of the dataset (Marron et al., 2014).

In Table 3, the chosen hyperparameters and their ranges can be found. The ranges for the number of estimators, max depth, and min samples leaf have been based on the research done by Daldır et al. (2021). However, the range for max depth has been slightly modified to search deeper; they used

9 nodes as a limit, while I increased it to 20. This was chosen because the data used in this study has more features and rows, theoretically needing to find more complex relationships. The problem that this could induce is overfitting on the training set and not generalizing well on the test set; this has to be taken into account during cross-validation on the train and validation sets. Moreover, the feature min sample split was added; it is similar to min sample leaf. It will increase or decrease the number of samples before a node is split; this impacts the depth of the tree with fewer points leading to a much complex and deeper tree, while more does the opposite.

Table 3: Hyperparameter Ranges/Values for Random Forests

Hyperparameter	**Ranges/Values**
max_depth	[3, 20], step 1
n_estimators	[150, 350], step 50
min_samples_leaf	[5, 9], step 1
min_samples_split	[2, 10], step 1

4.4.3 *XGBoost*

Like Random Forests, XGBoost (eXtreme Gradient Boosting) is an ensemble method that combines the outputs from multiple decision trees. However, there is a difference between these models. Random forests build their trees independently, and a final prediction is obtained by aggregating the individual trees; XGBoost builds its trees sequentially, where each tree corrects the errors made by the previous tree. This ensures it can learn from mistakes and find even more complex relationships within the data.

Moreover, XGBoost uses a Gradient Boosting algorithm. It aims to minimize the loss function and does this by iteratively adding decision trees. The weights of the misclassified data points are adjusted and thus improving upon previous trees. It also applies regularization techniques to prevent overfitting (T. Chen & Guestrin, 2016).

Lastly, and perhaps the most important, the model is scalable; it uses parallel processing and distributed computing, allowing it to be very efficient in its memory usage. Furthermore, unlike Random Forests from Scikit-Learn, XGBoost can run on GPUs, making it computationally efficient. Other than its efficiency, it has also been a proven method within the literature (Chakrabarty et al., 2019; Daldır et al., 2021; Hatıpoğlu et al., 2022; Manna et al., 2018). This makes the model a suitable candidate and a competitive model for TabNet.

The hyperparameters and their ranges that have been chosen can be found in Table 4. The n_estimators and max_depth hyperparameters were reused from the Random Forests algorithm so to be able to compare these models fairly. Other parameters like the learning_rate (eta), subsampling ratio of the training instances, the minimum sum of the instance weight in a child node, and the subsample ratio of a column have been based on previous research in the literature (Daldır et al., 2021; Hatıpoğlu et al., 2022).

Table 4: Hyperparameter Ranges/Values for XGBoost

Hyperparameter	**Ranges/Values**
max_depth	[3, 20], step 1
n_estimators	[150, 350], step 50
learning_rate	[5e-2, 1e-2, 15e-2, 1e-1, 2e-1]
subsample	[0.0, 1.0], log sampled
min_child_weight	[0, 20], step 1
colsample_bytree	[0.8, 1.0], log sampled

4.4.4 *TabNet*

TabNet, developed by Arık and Pfister (2021), is a Deep Neural Network (DNN) specifically designed to operate with tabular data. Its unique feature is the ability to work with unprocessed data, simplifying utilization while providing built-in interpretability through model-specific global and local explanations.

The TabNet architecture comprises sequential multi-steps (n_steps), where each step contributes to determining the features used, reminiscent of a decision tree. Data is divided into batches of size X (batch_size) serving as input. The model initiates by employing a Batch Normalization (BN) layer, calculating mean and variance over the input data, which is then fed into the Feature Transformer.

The Feature Transformer (FT) consists of N Gated Linear Units (GLU) blocks, each comprising three layers: Fully Connected (FC) layer, BN, and a GLU nonlinearity. For instance, when four GLU blocks are used, two are shared (n_shared), and two are independent, enhancing robustness and efficiency (Joseph et al., 2022). The input size for the FT is the number of features, and the output size is split into n_d + n_a, where n_d is the width of the decision dimension, and n_a is the width of the attention dimension. N_a serves as input for the Attentive Transformer (AT), and n_d serves as input for the ReLU function.

Subsequently, the AT processes the split input, composed of four layers: FC, BN, prior scales, and sparsemax. The prior scale indicates how much a feature has been used in prior steps, while sparsemax normalizes the coefficients. In the prior scale, the parameter gamma can be configured, where a higher gamma increases the reuse of features in the mask. The output size is the number of features, allowing for the reapplication of the mask, then the process starts anew (Arık & Pfister, 2021). Visual representation of this architecture can be found in Figure 7.

TabNet has been selected for this paper due to its demonstrated superiority over other high-performing models like XGBoost, CatBoost, and Random Forests (Joseph et al., 2022; McDonnell et al., 2023; Zhen & Zhu, 2023). These well-established models have shown leading results in the airplane delay domain. The objective is to assess whether TabNet can surpass or match the performance of these benchmarks.

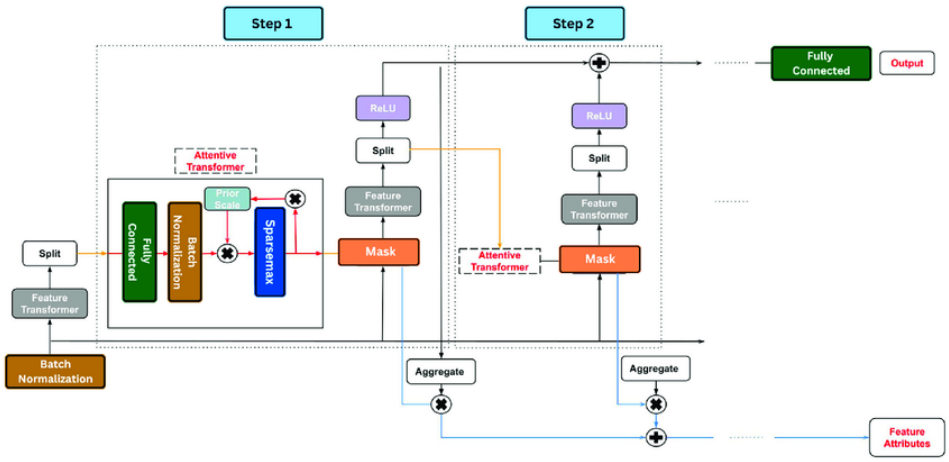

Figure 7: TabNet Architecture - Image by McDonnell et al. (2023), used under the CC BY 4.0 license, image belongs to the original author

For TabNet, the chosen hyperparameters and their ranges are detailed in Table 5. Some of these have been previously mentioned in the model explanation. Arık and Pfister (2021) provided recommendations for selecting hyperparameters and their ranges, advising against excessively high n_steps to prevent overfitting. Moreover, cautioning against a high value for n_d and n_a to again avoid overfitting. Batch sizes are suggested to be 1-10% of the total training dataset, if memory allows. Additional hyperparameters were selected based on the documentation for the PyTorch implementation of TabNet (Fischman, 2023).

Table 5: Hyperparameter Ranges/Values for TabNet

Hyperparameter	**Ranges/Values**
n_d	[8, 64], step 4
n_a	n_d = n_a
n_steps	[2, 10], step 1
gamma	[1.0, 2.0], step 0.01
n_shared	[1, 5], step 1
lambda_sparse	[1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1]
batch_size	44000
optimizer	Adam
learning_rate	1e-2

4.4.5 *Hyperparameter tuning*

The chosen models entail a wide array of hyperparameters that require tuning. Initially, a Random Search method was considered due to its computational efficiency in finding good hyperparameters. However, Bayesian Optimization proved to be a better fit for this problem as it finds more optimal parameters in less time. This method starts with a random guess, then finds the best parameters, and continues with these, thus making more informed decisions than Random Search in less time (Turner et al., 2021). Two prominent packages that employ this method are HyperOpt and Optuna. For this study, Optuna was chosen as it was deemed more user-friendly and easier to implement (Shekhar et al., 2021). Optuna utilizes a Tree-structured Parzen Estimator (TPE) algorithm, making it a suitable candidate if time is a crucial factor. Additionally, the package offers insightful visualizations, aiding in a deeper understanding of the performance of each model (Akiba et al., 2019).

Within Optuna, a hyperparameter N of trials can be selected. These trials involve evaluating a given objective function. To provide a fair assessment, the same N of trials will be used for all models. The optimal N was determined by running an XGBoost model and observing when it would stop improving. An N of 50 was selected, demonstrating that 20 trials should be sufficient to find the most optimal hyperparameters, thus reducing the time required for each model (see Figure 8 for the results).

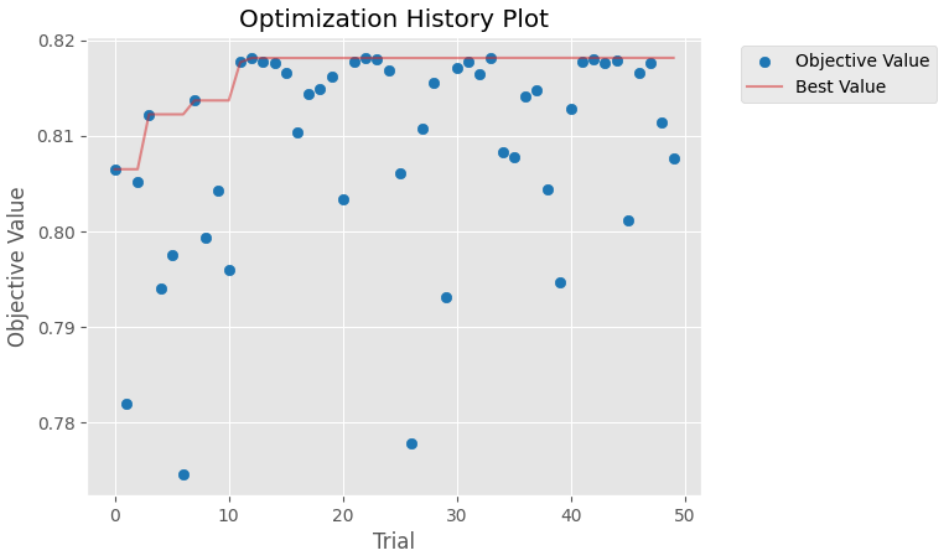

Figure 8: 50 Optuna Trials with XGBoost

To determine the best hyperparameters, a stratified k-fold cross-validation method was applied to the training set. A stratified approach was chosen to ensure each fold has the same proportion of the binary class labels, ensuring representativity while reducing variance (Prusty et al., 2022). A k-fold cross-validation method was chosen to tune the hyperparameters more optimally by exposing it to more data, allowing it to find the most optimal parameters while ensuring generalizability, robustness, and minimizing overfitting. Three was chosen as the number of K in the K-folds as recommended by Yadav and Shukla (2016). This choice was based on the large dataset, which necessitates a smaller number of K compared to a smaller dataset that might require a higher number for more data exposure to reduce bias.

4.5 *Class Imbalance methods*

As observed in Section 4.2, there is a class imbalance present in the binary target variables. This imbalance could impact the model's performance and accuracy. To address this, three different resampling methods will be tested to evaluate their impact on the performance of the four different models: oversampling, undersampling, and hybrid-sampling (a combination of over- and undersampling). All these methods are applied on the data-level. It should be noted that some of the chosen ensemble models are more resilient against class imbalance on an algorithm-level, particularly models like Random Forests and XGBoost (Maheshwari et al., 2017).

4.5.1 *Oversampling*

Several strategies can be applied to oversampling; in this study, we will apply an oversampling strategy to the minority class. This is chosen because the minority class is the one that we want to predict more accurately. Random Oversampling (ROS) will be applied to the data as the oversampling method. This method is computationally efficient as it randomly selects entries, with replacement, and adds them to the training set. The downside is that it might increase overfitting and training time of the models (Wongvorachan et al., 2023). Other methods like SMOTE and ADASYN are known for their high performance but require more computational resources and are therefore not used.

4.5.2 *Undersampling*

For undersampling, the majority class will be undersampled, decreasing the number of oversamples in the majority class and bringing both classes to an equal amount. The downside of this strategy is that it might remove valuable data important for the models, consequently decreasing accuracy (Maheshwari et al., 2017). Random Undersampling (RUS) will be applied to the training data as it is easy to implement and computationally efficient. Other samplers like Tomek-Links and Edited Nearest Neighbours have also been considered but proved to be computationally too expensive.

4.5.3 *Hybrid-Sampling*

Hybrid combines the upsides of both over and undersampling. It increases the minority class by oversampling but then decreases the size by undersampling, thus decreasing the risk of overfitting. The downside is that it does increase the training time for the models. Hybrid methods have been proven to never be worse than the base sampler and mostly improve accuracy (Seiffert et al., 2008). In this study, ROS and RUS will be applied to the training set to see if this improves the performance of the models. Other methods like SMOTE-Tomek or SMOTE-ENN have also been tested and tried but proved to be computationally too expensive for this study. ROS also proved to be as competitive as the previously stated hybrid methods (Batista et al., 2004).

4.6 *Evaluation methods*

The evaluation method plays a crucial role in assessing the models, as certain methods could provide a skewed image of their performance. The primary method for evaluating performance will be the Area Under the

ROC Curve (AUROC or AUCROC). Due to class imbalance, resilient methods are necessary, as metrics like Accuracy may give a distorted view. Research by Wardhani et al. (2019) demonstrated that AUROC is the least affected by class imbalance, making it the main evaluation metric in this study.

In addition to AUROC, the F1-Score will also be assessed. This value is used in related studies within the flight delay domain, allowing for comparisons with other research. Accuracy will also be reported for the same reason as the F1-score. However, it should be noted that accuracy can present a biased view due to a high true positive rate, which is more impactful for the majority class.

Finally, precision and recall values will be reported to assess the models in terms of false positives and false negatives. A higher recall is preferred, as False Negatives have more impact on passengers than False Positives. Identifying a plane as delayed when it is not provides relief, whereas failing to identify a delayed plane could be frustrating.

4.7 *Experiment setup*

In this section, the experiments will be detailed to enhance reproducibility and transparency, each linked to one of the three research questions in the same order.

4.7.1 *Experiment 1: Baselines and Tuned Models on Imbalanced Data*

Initially, baselines were established using four untuned models on imbalanced data after preprocessing. These baselines will be compared to assess the impact of tuning on the models. The four models were then re-evaluated on the imbalanced data after tuning using the predefined hyperparameter search spaces. The tuned models underwent evaluation using the predefined metrics and the Stratified K-Fold Cross-Validation method on the training set to identify the most and least influential hyperparameters. Optuna was utilized to find optimal parameters through an objective function containing the Stratified K-Fold Cross-validation and hyperparameter search spaces.

4.7.2 *Experiment 2: Balancing the Data and Retuning the Models*

In this experiment, all four models underwent three different types of sampling. Models were retuned on the training data using Optuna and the Stratified K-Fold Cross-Validation methods, considering potential shifts in optimal parameters due to data differences. ROS, RUS, and the hybrid

method (ROS+RUS) were applied to all four models, and results were evaluated based on the previously mentioned metrics.

4.7.3 *Experiment 3: Evaluating Errors per Airline by the Best Model*

The best-performing model with optimal hyperparameters was chosen, and its predictions were analyzed based on the number of prediction mistakes per airline. These errors were plotted and compared while considering the total number of flights per airline.

5 RESULTS

5.1 *Experiment 1 results*

Firstly, the results of both the baseline and subsequently tuned models are provided. The outcomes on the imbalanced data can be found in Table 6, utilizing the metrics described in Section 4.6. The scores represent the performance on the testing data by the models, with the best-performing scores highlighted. Optimal hyperparameters identified through tuning are detailed in Appendix C.

Several noteworthy observations arise, with the most conspicuous being that TabNet did not enhance its performance through tuning but, in fact, performed worse. A comparison of all confusion matrices in Figures 9 and 15 (Appendix D) reveals that the untuned model excels at predicting non-delays and makes fewer false positive mistakes, resulting in higher precision. Conversely, for the tuned model, the situation is reversed; it exhibits a lower false negative rate and better true negative predictions, hence achieving higher precision. However, a higher score does not necessarily indicate superior delay prediction, as the model excels in predicting non-delays, emphasizing the need for a focus on predicting delays.

LR does not appear to improve or decline in performance when left untuned or tuned. A primary reason for this could be attributed to the unsuitability of regression models for this problem, primarily because they may struggle to identify more complex relationships within the data, especially when the problem is not linearly separable. RF and XGBoost seem to be the top performers, with XGBoost benefiting the most from tuning, displaying a 2.8% improvement in AUROC and achieving the highest overall scores when tuned.

Table 6: Baseline and Tuned Model Performance Metrics

Model	Configuration	AUROC	Accuracy	F1-Score	Recall	Precision
LR	Untuned baseline	0.739	0.841	0.473	0.377	0.634
	Tuned	0.739	0.841	0.473	0.377	0.634
XGB	Untuned baseline	0.794	0.849	0.480	0.371	0.682
	Tuned	**0.822**	**0.858**	**0.526**	**0.416**	**0.713**
RF	Untuned baseline	0.797	0.850	0.488	0.378	0.688
	Tuned	0.798	0.850	0.488	0.378	0.687
TabNet	Untuned baseline	0.774	0.846	0.473	0.367	0.666
	Tuned	0.760	0.843	0.480	0.384	0.640

In conclusion, all models exhibit higher precision and lower recallability. This is evident in the confusion matrix for the tuned models in Figure 9, with a higher number of false negatives and a lower number of false positives. The same pattern is observed in the confusion matrix for the untuned data in Figure 15 (Appendix D). This suggests that the models inherently struggle with classifying delays, often opting to classify 'no delay.' This trend could be attributed to the class imbalance, as the models have less data to assist in classifying delayed flights. As stated in Section 4.6, a higher recall is preferred. By that metric and the AUROC main metric, the tuned XGBoost model will be considered the best-performing model in this experiment.

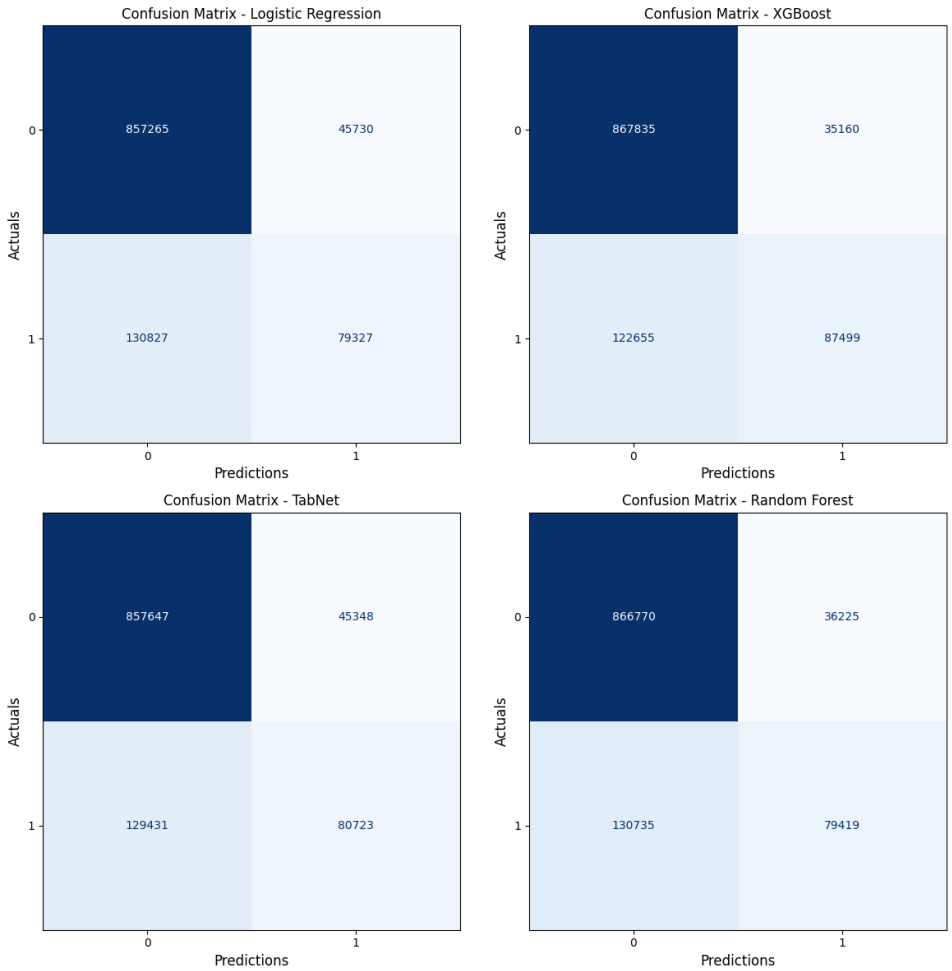

Figure 9: Four confusion matrices of the tuned models on imbalanced data

5.2 *Experiment 2 results*

The results of the models on the test data with various balancing techniques are presented in Table 7. An immediate distinction between this experiment and the first one is that all recall scores have increased, while precision scores have decreased. This means that, as expected, resampling enhanced the recallability of the minority class (1), leading to more accurate predictions. However, it came at a trade-off. As illustrated in the confusion matrices in Figures 10, 17, and 18 (Appendix D), the models exhibited lower false negative values than in experiment 1 and higher false positive values, resulting in an increased recall and decreased precision.

Once again, TabNet did not improve in terms of AUROC with resampling; it remained below the untuned results of experiment 1. Analyzing

the hyperparameter importance in Figure 22 (Appendix E), as provided by Optuna, the number of steps appeared to be the most critical parameter when resampling. This could be a limiting factor in finding the optimal configuration, explaining the worse performance. It is known that TabNet requires more tuning to perform well (McDonnell et al., 2023).

LR also showed no improvement or deterioration in results when resampled. Similar to the other models, recall increased, and precision decreased, confirming the belief that the model struggles to identify more complex relations when determining if a plane is delayed. Furthermore, when comparing the found feature importance in the Optuna studies (Figure 20 in Appendix E), it was determined that the solver parameter was the most influential. Despite this, according to the documentation (Pedregosa et al., 2011), it should have yielded the most optimal performance.

RF and XGBoost once again emerged as the best-performing models in terms of AUROC and recall. Reflecting on experiment 1, XGBoost decreased in AUROC, while RF showed little to no improvement. Examining the hyperparameter importance as found by Optuna (Figures 21 and 23 in Appendix E), both models suggested that the depth of the trees contributed the most to their accuracy. Moreover, for XGBoost, the subsample parameter was deemed influential in performance, reducing overfitting in the model.

Table 7: Model Performance Metrics with Resampling Techniques

Model	Resampling Technique	AUROC	Accuracy	F1-Score	Recall	Precision
LR	ROS	0.740	0.797	0.481	0.499	0.465
	RUS	0.740	0.797	0.481	0.499	0.465
	ROS+RUS (Hybrid)	0.740	0.797	0.481	0.499	0.465
XGB	ROS	**0.819**	0.813	0.563	0.640	0.503
	RUS	0.817	0.788	0.547	**0.679**	0.458
	ROS+RUS (Hybrid)	**0.819**	**0.816**	**0.566**	0.635	**0.510**
RF	ROS	0.798	0.814	0.543	0.584	0.507
	RUS	0.795	0.798	0.532	0.609	0.473
	ROS+RUS (Hybrid)	0.798	0.814	0.543	0.584	0.506
TabNet	ROS	0.768	0.773	0.496	0.593	0.426
	RUS	0.746	0.698	0.449	0.653	0.342
	ROS+RUS (Hybrid)	0.767	0.790	0.501	0.559	0.454

In summary, resampling significantly enhanced the recallability of all models, justifying the effectiveness of the procedure. However, it had an impact on precision, and this decline may be attributed to the chosen sampling methods, potentially introducing more randomness and noise. It is important to note a higher number of false positives , as evident in Figure 10 (and Figures 17 and 18 in Appendix D). This implies that all models exhibited a preference for classifying a flight as delayed rather than non-delayed. Both experiments were affected by this skew, leading to the classification of either too many false positives or false negatives.

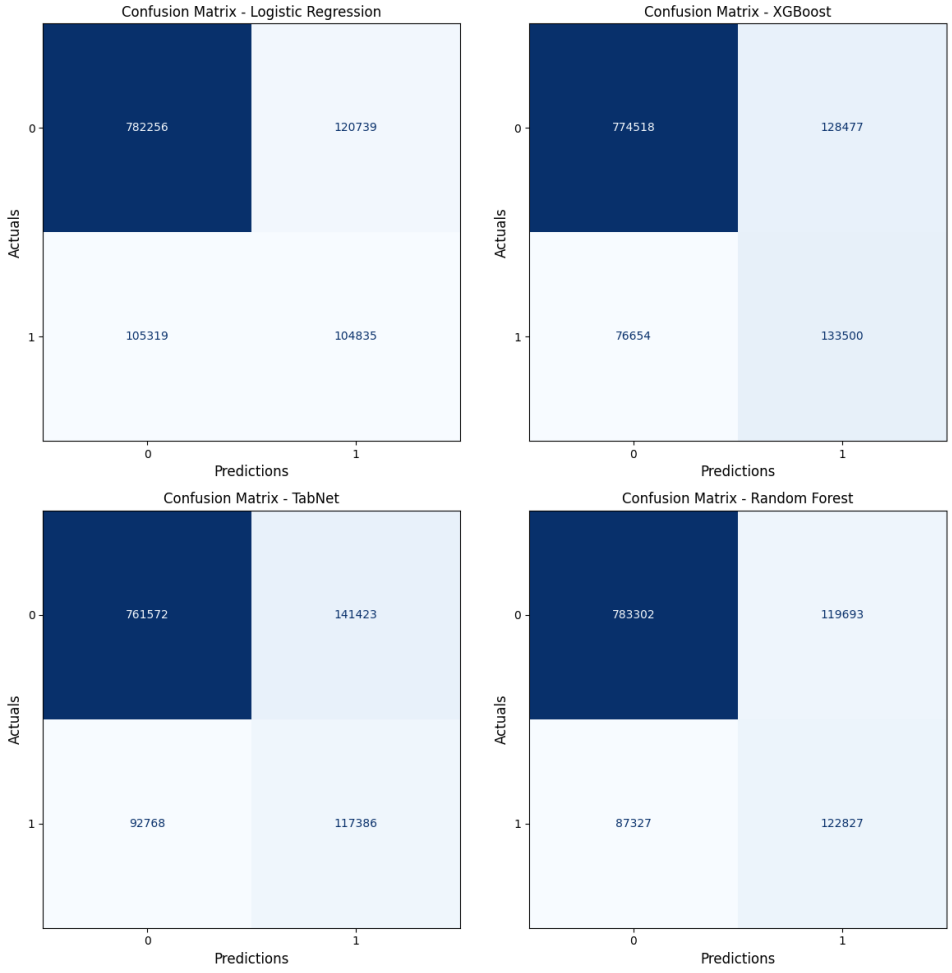

Figure 10: Four confusion matrices of the tuned models on ROS+RUS resampled data

Finally, the ROS+RUS resampled XGBoost model is considered the best-performing model in both experiments and will be utilized to formulate the conclusions of experiment 3. The primary contributing factor is the higher Area Under the Receiver Operating Characteristic (AUROC) score,

coupled with an increased recall score and a well-balanced precision score. Although not necessarily the highest across all metrics, the model generally demonstrates greater accuracy in predicting delays without compromising performance in predicting non-delays.

5.3 *Experiment 3 results*

The prediction results from the hybrid-sampled XGBoost model will offer insights into the accuracy of delay predictions for each airline and identify those that are less predictable. Figure 11 presents the outcomes of this analysis. Prediction errors, where the model made incorrect predictions, were tallied and grouped by airlines. Subsequently, these values were contrasted with the total number of flights for each airline, providing a fair assessment of performance. Airlines with higher numbers of prediction errors might also experience an increase in total flights. To ensure fairness in this evaluation, the prediction errors were divided by the total number of flights, yielding a percentage ratio of errors relative to total flights. These percentages are depicted atop the bars in the figure.

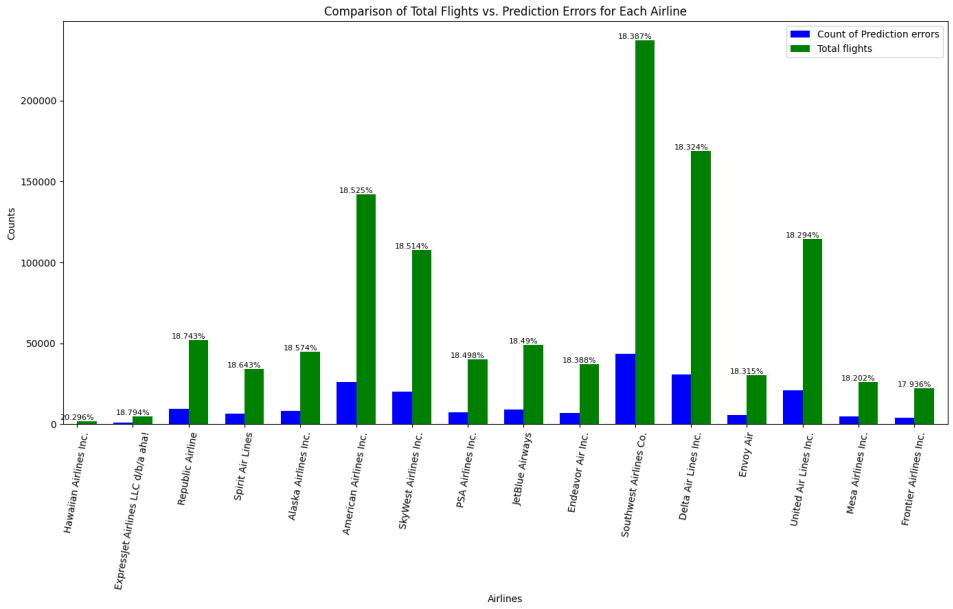

Figure 11: Comparison of Total Flights vs. Prediction Errors for Each Airline with relative error percentages at the top of each bar

An intriguing observation is that Hawaiian Airlines, despite having the fewest total flights, proves to be the most challenging to predict, with a percentage error of 20.296%. This could be attributed to the limited available data. However, major airlines like American Airlines or Delta

do not necessarily exhibit greater predictability. Frontier Airlines made the fewest prediction errors, with a rate of 17.936%, indicating a notable difference of 2.36% between the two.

6 DISCUSSION

The primary objective of this study was to forecast flight delays using Logistic Regression, Random Forests, TabNet, and XGBoost while examining the impact of resampling techniques. Additionally, an investigation into airline comparison based on prediction errors from the best model aimed to unveil any notable differences in delay predictability among airlines. The results indicated that resampling increased recall, enhancing the ability to classify delays. However, this improvement came at the cost of decreased precision compared to when imbalanced data was utilized. Overall, the hybrid-sampled XGBoost model emerged as the best performer with an AUROC of 0.819. Notably, a distinct difference of 2.36% in terms of prediction errors per airline was observed.

6.1 *Results Discussion*

In section 3.4, the study by Dand et al. (2019) was identified as the State-of-the-Art, achieving an AUROC score of 0.62 using Naïve Bayes. In comparison, our research achieved a significantly higher AUROC score, demonstrating enhanced accuracy in predicting flight delays. A critical distinction lies in the inclusion of daily weather data in our research, an aspect not included by Dand et al. (2019). The feature importance analysis of the best model in Figure 12 highlights the considerable impact of weather conditions, with temperature, wind and precipitation playing pivotal roles. Additionally, the engineered feature assessing the delay status of a previous flight significantly influenced the decision-making process of the model. While Section 4.3.2 identified a moderate correlation of this feature with the target variable, it now underscores its influence in predicting flights. Comparing these crucial features with those identified by Dand et al. (2019), similarities emerge, including month, departure time, and arrival time, all recognized as top 10 influential features.

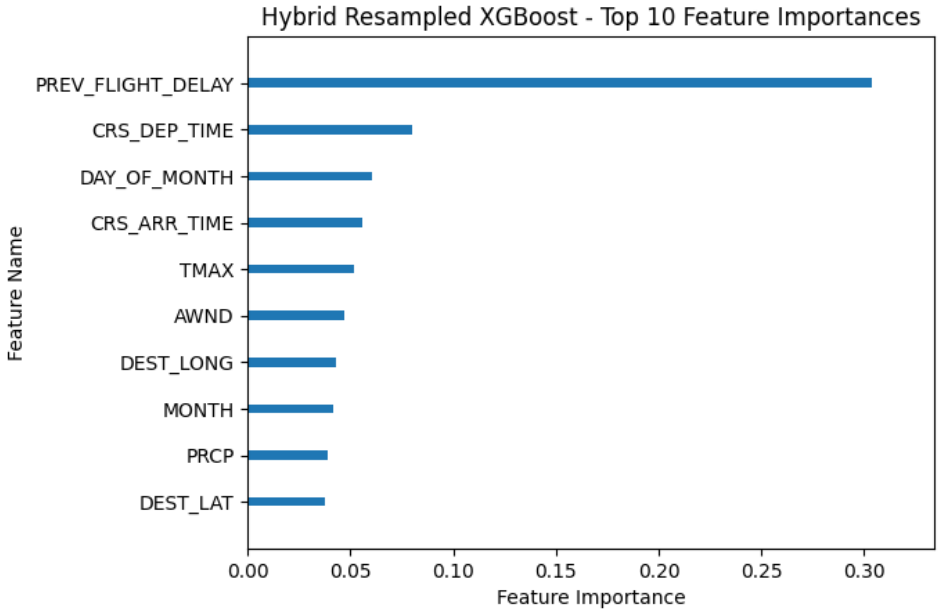

Figure 12: Top 10 Feature importance for the XGBoost model on the Hybrid resampled data

Contrary to literature suggesting TabNet's outperformance of established models, our study found that TabNet was surpassed by both RF and XGBoost. A closer examination of Figure 13, illustrating the feature importance of the hybrid-sampled TabNet model, reveals a distinct divergence. TabNet prioritized airline carriers as important features, a focus not shared by other models. This disparity could be a contributing factor to its suboptimal performance. As acknowledged by McDonnell et al. (2023), TabNet demands longer training times, a constraint that influenced its performance in our study, given the limited time and computational resources available. This limitation, along with the complexity of the TabNet model, likely contributed to its inferior performance compared to XGBoost, which requires less training time and resources.

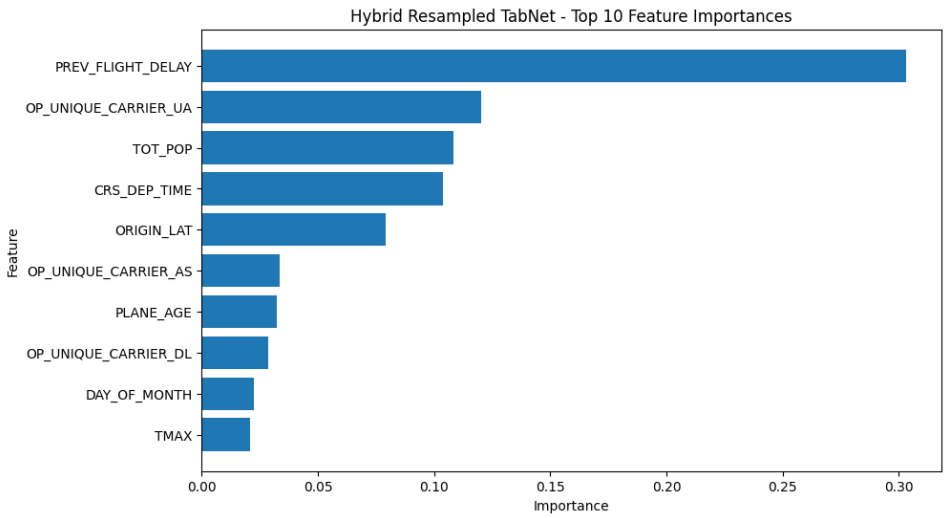

Figure 13: Top 10 Feature importance for the TabNet model on the Hybrid resampled data

6.2 *Limitations*

A prominent limitation identified in this study was the constraint on computational resources and model run-time, particularly for TabNet. The number of epochs for TabNet had to be set at three due to the associated increase in run-time, which was not feasible within the study's constraints. Additionally, while SMOTE and its variants showed promise in the literature, implementing these methods on our dataset proved impractical due to excessively long run-times, particularly with SMOTE oversampling alone taking 45 minutes for a single fold. This process had to be repeated three times and then an additional 20 times for Optuna optimization, rendering it unfeasible in the scope of this research. Lastly, some studies referenced in the literature used hourly weather data, theoretically offering more accurate insights. Unfortunately, for our study, only daily weather data near airfields from the NOAA was available.

6.3 *Scientific and Societal Relevance*

In terms of scientific contributions, this study suggests that TabNet may not be a suitable candidate for this type of problem or would require additional resources to outperform established models. Furthermore, the inclusion of the feature 'previous flight delayed' had a positive impact on the predictability of delays. Future research could incorporate this

feature and investigate its impact on existing studies, potentially enhancing predictability while considering the risk of overfitting.

Regarding social relevance, the best-performing model demonstrates efficacy across general airports and flights. This implies the feasibility of a single model for all flights and airports, without the need for optimization for specific flights or airports. Smaller airports could employ the same model used by larger airports, achieving comparable results without substantial resource requirements. This approach may even allow for resource pooling, thereby enhancing the traveler experience in both larger and smaller airports, reducing delays, and minimizing emissions.

6.4 *Future Work*

Regrettably, TabNet fell short in competing with established methods. However, TabPFN (Tabular Prior-Data Fitted Network), a novel deep learning model, has proven highly competitive (Hollmann et al., 2022). Requiring minimal hyperparameter tuning or data preprocessing, TabPFN delivers State-of-the-Art results. Exploring this model as a competitor to established RF and XGBoost models is recommended. Additionally, a study by Zhen and Zhu (2023) applied an ensemble learning approach by combining TabNet with other high-performing models, resulting in a 7% overall performance increase. This method holds promise for improving flight delay predictions through the combination of high-performing models. As mentioned earlier, only daily weather data was utilized. A potential direction for future research could involve incorporating hourly data to enhance model performance. This study was confined to 2019 data, but extending the timeframe could further enhance accuracy, albeit at the cost of increased computational resources. Finally, the significant impact of COVID-19 on the airline industry, with flights returning to pre-COVID numbers in the summer of 2022 (Sun et al., 2023), opens avenues for an interesting follow-up study. Comparing pre- and post-COVID flight prediction delays could provide valuable insights.

7 CONCLUSION

Air travel serves as one of the primary modes of transportation for intercontinental and intra-city travel. Despite its convenience, air travel is not without imperfections and is susceptible to delays. These delays impact not only passengers and airlines but also have environmental implications. This thesis seeks to mitigate these delays and demonstrate the effectiveness of a general predictive model for delays. Such a model could be implemented in smaller airports lacking the resources to develop their own

models, thereby reducing emissions, passenger dissatisfaction, and airline costs.

The primary research question guiding this study was: *To what extent can U.S. flight delays be predicted on imbalanced data using Machine/Deep learning models, and how do airlines differ in terms of the predictability of delays?* In conclusion, XGBoost emerged as the most accurate model for predicting delays, achieving an AUROC score of 0.819. Hybrid resampling demonstrated an improvement in recallability for predicting delays, emphasizing the significance of resampling. Furthermore, weather exerted a substantial influence on predictability, even with a larger dataset. While daily weather data was utilized, future studies should consider implementing hourly weather data to enhance predictability. The newly engineered 'previous flight delayed?' binary feature proved highly influential in predicting delays, suggesting its potential value in other studies to enhance accuracy.

Lastly, a notable distinction in the predictability of airlines was observed, with a 2.36% difference between the most and least predictable airlines. Increasing the dataset, particularly historic data, could potentially narrow this gap. For instance, Hawaiian Airlines, the least predictable airline, also had the fewest datapoints. Overall, this study advanced the state-of-the-art and introduced new influential methods for the flight delay prediction domain.

REFERENCES

Airlines For America. (2023). *U.s. passenger carrier delay costs*. https://www.airlines.org/dataset/u-s-passenger-carrier-delay-costs/

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2623–2631. https://doi.org/10.1145/3292500.3330701

Arık, S., & Pfister, T. (2021). Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*, 6679–6687. https://doi.org/10.1609/AAAI.V35I8.16826

Ayer, T., Chhatwal, J., Alagoz, O., Kahn, C. E., Woods, R. W., & Burnside, E. S. (2010). Comparison of logistic regression and artificial neural network models in breast cancer risk estimation1. *https://doi.org/10.1148/rg.301095057*, *30*, 13–22. https://doi.org/10.1148/RG.301095057

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, *6*, 20–29. https://doi.org/10.1145/1007730.1007735

Bilogur, A. (2018). Missingno: A missing data visualization suite. *Journal of Open Source Software*, *3*(22), 547. https://doi.org/10.21105/joss.00547

Bisong, E. (2019). Logistic regression. *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, 243–250. https://doi.org/10.1007/978-1-4842-4470-8_20

Boggavarapu, R., Agarwal, P., & Kumar, D. H. R. (2019). Aviation delay estimation using deep learning. *2019 4th International Conference on Information Systems and Computer Networks, ISCON 2019*, 689–693. https://doi.org/10.1109/ISCON47742.2019.9036276

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32. https://doi.org/10.1023/A:1010933404324/METRICS

Buchholz, K. (2023). *Delayed flights - the new post-pandemic normal?* https://www.statista.com/chart/29098/share-of-late-arrivals-by-north-american-airlines/

Bureau of Transportation Statistics. (2023a). *B-43 inventory*. Bureau of Transportation Statistics. https://www.transtats.bts.gov/Fields.asp?gnoyr_VQ=GEH

Bureau of Transportation Statistics. (2023b). *Full year 2022 u.s. airline traffic data*. https://www.bts.gov/newsroom/full-year-2022-us-airline-traffic-data

Bureau of Transportation Statistics. (2023c). *On-time performance*. Bureau of Transportation Statistics. https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGJ&QO_fu146_anzr=bo-gvzr

Bureau of Transportation Statistics. (2023d). *On-time performance - reporting operating carrier flight delays at a glance*. https://www.transtats.bts.gov/homedrillchart.asp

Chakrabarty, N. (2019). A data mining approach to flight arrival delay prediction for american airlines. *IEMECON 2019 - 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference*, 102–107. https://doi.org/10.1109/IEMECONX.2019.8876970

Chakrabarty, N., Kundu, T., Dandapat, S., Sarkar, A., & Kole, D. K. (2019). Flight arrival delay prediction using gradient boosting classifier. *Advances in Intelligent Systems and Computing*, *813*, 651–659. https://doi.org/10.1007/978-981-13-1498-8_57/TABLES/3

Chen, J., & Li, M. (2019). Chained predictions of flight delay using machine learning. https://doi.org/10.2514/6.2019-1661

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 785–794. https://doi.org/10.1145/2939672.2939785

Choi, S., Kim, Y. J., Briceno, S., & Mavris, D. (2016). Prediction of weather-induced airline delays based on machine learning algorithms. *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, *2016-December*. https://doi.org/10.1109/DASC.2016.7777956

Daldır, I., Tosun, N., & Tosun, Ö. (2021). Performance evaluation of machine learning techniques on flight delay prediction. *Lecture Notes on Data Engineering and Communications Technologies*, *76*, 165–173. https://doi.org/10.1007/978-3-030-79357-9_16

Dand, A., Saeed, K., & Yildirim, B. (2019). Prediction of airline delays using faa data prediction of airline delays based on machine learning algorithms. *25th Americas Conference on Information Systems*.

Fischman, S. (2023). *Pytorch tabnet*. https://pypi.org/project/pytorch-tabnet/

Hancock, J. T., & Khoshgoftaar, T. M. (2020). Survey on categorical data for neural networks. *Journal of Big Data*, *7*, 1–41. https://doi.org/10.1186/S40537-020-00305-W/FIGURES/4

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020).

Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hatıpoğlu, I., Tosun, Ö., & Tosun, N. (2022). Flight delay prediction based with machine learning. *Logforum*, *18*, 97–107. https://doi.org/10.17270/J.LOG.2022.655

Hollmann, N., Müller, S., Eggensperger, K., & Hutter, F. (2022). Tabpfn: A transformer that solves small tabular classification problems in a second. https://arxiv.org/abs/2207.01848v6

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

IP2Location. (2023). *Iata geospatial coordinates*. IP2Location. https://github.com/ip2location/ip2location-iata-icao

Jiang, Y., Liu, Y., & Liu, D. (2020). Applying machine learning to aviation big data for flight delay prediction. *International Symposium on Dependable, Autonomic and Secure Computing (DASC)*. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00114

Jordahl, K., den Bossche, J. V., Fleischmann, M., Wasserman, J., McBride, J., Gerard, J., Tratner, J., Perry, M., Badaracco, A. G., Farmer, C., Hjelle, G. A., Snow, A. D., Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur, A., … Leblanc, F. (2020, July). *Geopandas/geopandas: V0.8.1* (Version v0.8.1). Zenodo. https://doi.org/10.5281/zenodo.3946761

Joseph, L. P., Joseph, E. A., & Prasad, R. (2022). Explainable diabetes classification using hybrid bayesian-optimized tabnet architecture. *Computers in Biology and Medicine*, *151*, 106178. https://doi.org/10.1016/J.COMPBIOMED.2022.106178

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, *18*(17), 1–5. http://jmlr.org/papers/v18/16-365.html

Maheshwari, S., Jain, R. C., & Jadon, R. S. (2017). A review on class imbalance problem: Analysis and potential solutions. https://doi.org/10.20943/01201706.4351

Manna, S., Biswas, S., Kundu, R., Rakshit, S., Gupta, P., & Barman, S. (2018). A statistical approach to predict flight delay using gradient boosted decision tree. *ICCIDS 2017 - International Conference on Computational Intelligence in Data Science, Proceedings*, *2018-January*, 1–5. https://doi.org/10.1109/ICCIDS.2017.8272656

Marron, D., Bifet, A., & Morales, G. D. F. (2014). Random forests of very fast decision trees on gpu for mining evolving big data streams.

Frontiers in Artificial Intelligence and Applications, *263*, 615–620. https://doi.org/10.3233/978-1-61499-419-0-615

McDonnell, K., Murphy, F., Sheehan, B., Masello, L., & Castignani, G. (2023). Deep learning in insurance: Accuracy and model interpretability using tabnet. *Expert Systems with Applications*, *217*. https://doi.org/10.1016/J.ESWA.2023.119543

McKinney, W., et al. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, *445*, 51–56.

National Oceanic and Atmospheric Administration. (2023). *Climate data online*. National Oceanic and Atmospheric Administration. https://www.ncdc.noaa.gov/cdo-web/datasets

Pedersen, A. B., Mikkelsen, E. M., Cronin-Fenton, D., Kristensen, N. R., Pham, T. M., Pedersen, L., & Petersen, I. (2017). Missing data and multiple imputation in clinical epidemiological research. *Clinical Epidemiology*, *9*, 157. https://doi.org/10.2147/CLEP.S129785

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Prusty, S., Patnaik, S., & Dash, S. K. (2022). Skcv: Stratified k-fold cross-validation on ml classifiers for predicting cervical cancer. *Frontiers in Nanotechnology*, *4*, 972421. https://doi.org/10.3389/FNANO.2022.972421/BIBTEX

Rahul, R., Kameshwari, S., & Kumar, R. P. (2022). Flight delay prediction using random forest classifier. *Lecture Notes in Electrical Engineering*, *783*, 67–72. https://doi.org/10.1007/978-981-16-3690-5_7

Ryerson, M. S., Hansen, M., & Bonn, J. (2014). Time to burn: Flight delay, terminal efficiency, and fuel consumption in the national airspace system. *Transportation Research Part A: Policy and Practice*, *69*, 286–298. https://doi.org/10.1016/J.TRA.2014.08.024

Seiffert, C., Khoshgoftaar, T. M., & Hulse, J. V. (2008). Hybrid sampling for imbalanced data. *2008 IEEE International Conference on Information Reuse and Integration, IEEE IRI-2008*, 202–207. https://doi.org/10.1109/IRI.2008.4583030

Shekhar, S., Bansode, A., & Salim, A. (2021). A comparative study of hyper-parameter optimization tools. *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering, CSDE 2021*. https://doi.org/10.1109/CSDE53843.2021.9718485

Stone, M. (2018). Impact of delays and cancellations on travel from small community airports. *Tourism and Hospitality Research*, *18*, 214–228.

https://doi.org/10.1177/1467358416637252/ASSET/IMAGES/LARGE/10.1177_1467358416637252-FIG4.JPEG

Sun, X., Wandelt, S., & Zhang, A. (2023). A data-driven analysis of the aviation recovery from the covid-19 pandemic. *Journal of Air Transport Management*, *109*, 102401. https://doi.org/10.1016/J.JAIRTRAMAN.2023.102401

Thiagarajan, B., Srinivasan, L., Sharma, A. V., Sreekanthan, D., & Vijayaraghavan, V. (2017). A machine learning approach for prediction of on-time performance of flights. *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, *2017-September*. https://doi.org/10.1109/DASC.2017.8102138

Turner, R., Eriksson, D., Mccourt, M., Kiili, J., Xu, V. Z., Escalante, H. J., & Hofmann, K. (2021, August). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. https://proceedings.mlr.press/v133/turner21a.html

U.S. Census Bureau. (2023). *Us cities demographics*. U.S. Census Bureau. https://public.opendatasoft.com/explore/dataset/us-cities-demographics/information/

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.

van Buuren, S., & Groothuis-Oudshoorn, K. (2011). Mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, *45*, 1–67. https://doi.org/10.18637/JSS.V045.I03

Wang, Y., Cao, L., Hao, D., al -, Kaur, N., Madan, J., A, M. K., & Ding, Y. (2017). Predicting flight delay based on multiple linear regression. *IOP Conference Series: Earth and Environmental Science*, *81*, 012198. https://doi.org/10.1088/1755-1315/81/1/012198

Wardhani, N. W. S., Rochayani, M. Y., Iriany, A., Sulistyono, A. D., & Lestantyo, P. (2019). Cross-validation metrics for evaluating classification performance on imbalanced data. *2019 International Conference on Computer, Control, Informatics and its Applications: Emerging Trends in Big Data and Artificial Intelligence, IC3INA 2019*, 14–18. https://doi.org/10.1109/IC3INA48034.2019.8949568

Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Wongvorachan, T., He, S., & Bulut, O. (2023). A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining. *Information 2023, Vol. 14, Page 54*, *14*, 54. https://doi.org/10.3390/INFO14010054

Yadav, S., & Shukla, S. (2016). Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification.

Proceedings - 6th International Advanced Computing Conference, IACC 2016, 78–83. https://doi.org/10.1109/IACC.2016.25

Zhen, Y., & Zhu, X. (2023). An ensemble learning approach based on tabnet and machine learning models for cheating detection in educational tests. *Educational and Psychological Measurement*, *0*. https://doi.org/10.1177/00131644231191298

APPENDIX A

Red columns have been omitted before merging, yellow columns have been omitted after preprocessing, green are present in the final dataset

Feature	Description	Origin	Datatype
MONTH	1 = January ..., 12 = December	BTS	int
DAY_OF_MONTH	Day of the month [1, 31]	BTS	int
DAY_OF_WEEK	1 = Monday ..., 7 = Sunday	BTS	int
OP_UNIQUE_CARRIER	2 character categorical variable used to identify the airline, see data files for further definition	BTS	category
TAIL_NUM	Unique number of an aircraft, combination of letters and numbers	BTS	category
ORIGIN_AIRPORT_ID	Unique airport id, used to identify the airport	BTS	int
ORIGIN	Unique IATA origin airport abbreviation	BTS	category
ORIGIN_CITY_NAME	City name abbreviation	BTS	category
DEST	Unique IATA destination airport abbreviation	BTS	category
CRS_DEP_TIME	4 digit planned departure time	BTS	int
DEP_TIME	Actual departure time, 4 digits	BTS	int
DEP_DELAY_NEW	Departure delay in minutes	BTS	int
DEP_DEL15	0=no delay, 1=delay, per BTS definition of delayed when a flight leaves 15 minutes later then planned	BTS	bool
CRS_ARR_TIME	4 digit planned arrival time	BTS	int
ARR_TIME	Actual arrival time, 4 digits	BTS	int
DISTANCE_GROUP	Miles between ORIGIN and DESTINATION, grouped together by integers, see data files for further definition	BTS	int
MANUFACTURE_YEAR	Used to create AIRPLANE_AGE	BTS	int
N_OF_SEATS	Number of seats on a plane	BTS	int
AWND	Maximum wind speed that day in Miles per Hour	NOAA	float
PRCP	Precipitation that day in inches	NOAA	float
SNOW	Snowfall that day in inches	NOAA	float
SNWD	Depth of snow that day in inches	NOAA	float
TMAX	Maximum temperature that day in Fahrenheit	NOAA	float
MEDIAN_AGE	Median age per origin city	USCB	float
TOT_POP	Total population per origin city	USCB	float
AVG_HOUSEHOLD_SIZE	Average household size per origin city	USCB	float
ORIGIN_LAT	Latitude for origin airport, [-90, 90]	ip2location	float
ORIGIN_LONG	Longitude for origin airport, [-180, 180]	ip2location	float
DEST_LAT	Latitude for destination airport, [-90, 90]	ip2location	float
DEST_LONG	Longitude for destination airport, [-180, 180]	ip2location	float

Figure 14: Overview of features used in final dataset and omitted features

APPENDIX B

Hardware:

- Google Colab+:
  - T4 GPU's 15 GB RAM
  - 51 GB RAM
- Personal device:
  - Intel i5-8250U
  - 8 GB RAM

Software/Packages:

- Visual Studio Code 1.84.2
- Python 3.10.11 (Van Rossum & Drake, 2009)
- Optuna 3.4.0 (Akiba et al., 2019)
- Pytorch-TabNet 4.1.0 (Fischman, 2023)
- Pandas 1.5.3 (McKinney et al., 2010)
- Numpy 1.23.5 (Harris et al., 2020)
- Scikit-Learn 1.2.1 (Pedregosa et al., 2011)
- Imblearn 0.10.1 (Lemaître et al., 2017)
- Matplotlib 3.6.3 (Hunter, 2007)
- Missingno 0.5.2 (Bilogur, 2018)
- Geopandas 0.14.0 (Jordahl et al., 2020)
- Seaborn 0.12.2 (Waskom, 2021)
- XGBoost 2.0.0 (T. Chen & Guestrin, 2016)

APPENDIX C

Logistic Regression

Table 8: Hyperparameter Values for Logistic Regression Models

Hyperparameters	Tuned	ROS	RUS	ROS+RUS
C	0.0123	0.0123	1.2542	0.0123
Solver	Saga	Saga	Sag	Saga
Penalty	None	None	None	None

XGBoost

Table 9: Hyperparameter Values for XGBoost Models

Hyperparameters	Tuned	ROS	RUS	ROS+RUS
n_estimators	350	200	350	300
learning_rate	0.05	0.05	0.05	0.05
max_depth	16	16	16	16
subsample	0.9951	0.8180	0.9952	0.9964
colsample_bytree	0.8668	0.8949	0.8669	0.8643
min_child_weight	8	17	8	16

Random Forests

Table 10: Hyperparameter Values for Random Forests Models

Hyperparameters	Tuned	ROS	RUS	ROS+RUS
max_depth	20	20	20	20
min_samples_leaf	5	8	8	8
min_samples_split	6	4	4	4
n_estimators	250	300	300	300

TabNet

Table 11: Hyperparameter Values for TabNet Models

Hyperparameters	**Tuned**	**ROS**	**RUS**	**ROS+RUS**
N_d	64	64	64	24
N_steps	4	4	2	2
Gamma	1.78	1.23	1.4	1.48
N_shared	1	4	5	4
lambda_sparse	0.001	0.0001	1e-06	0.0001

APPENDIX D

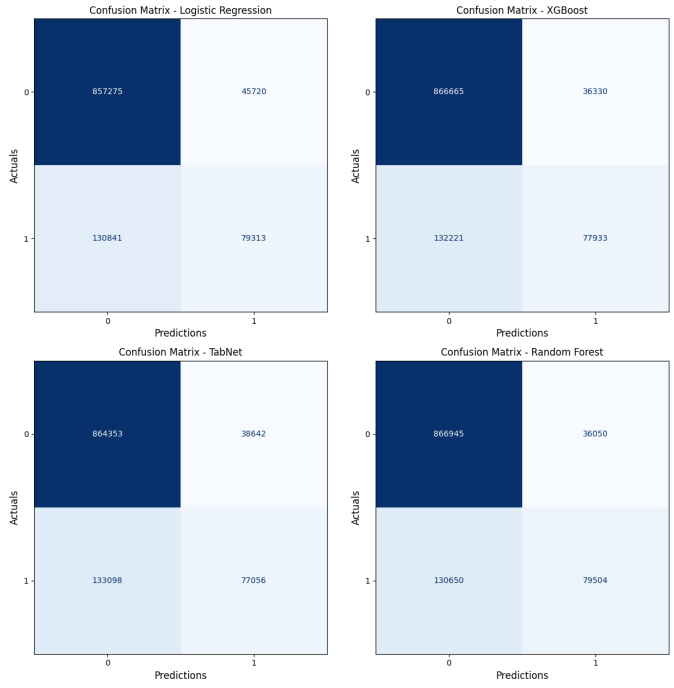

Figure 15: Four confusion matrices of the baseline untuned models on imbalanced data

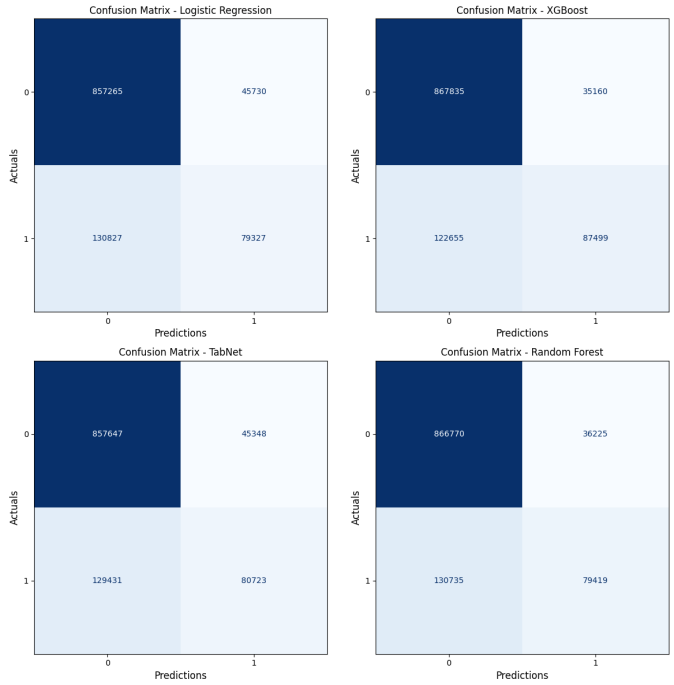

Figure 16: Four confusion matrices of the tuned models on imbalanced data

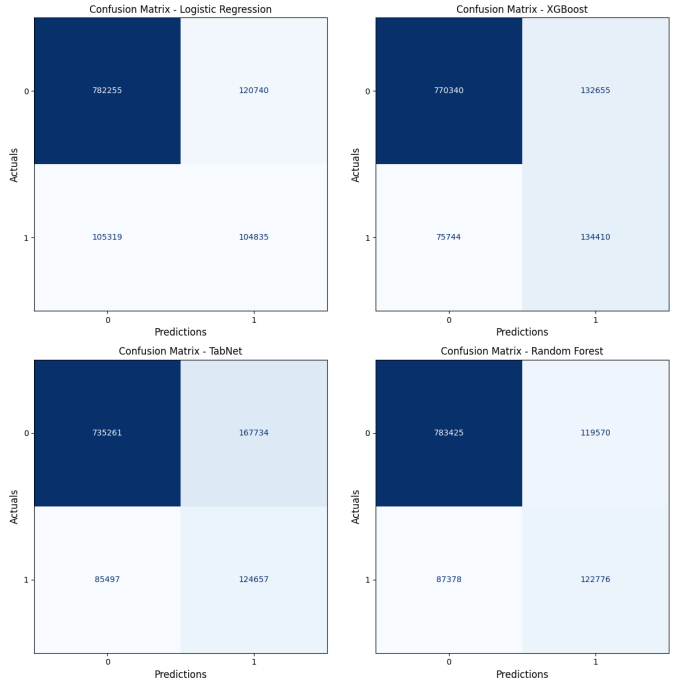

Figure 17: Four confusion matrices of the tuned models on ROS resampled data

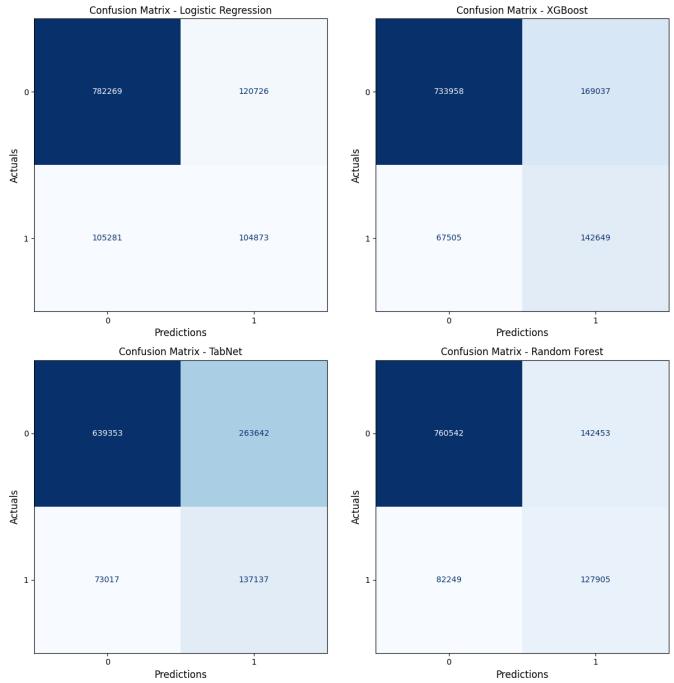

Figure 18: Four confusion matrices of the tuned models on RUS resampled data

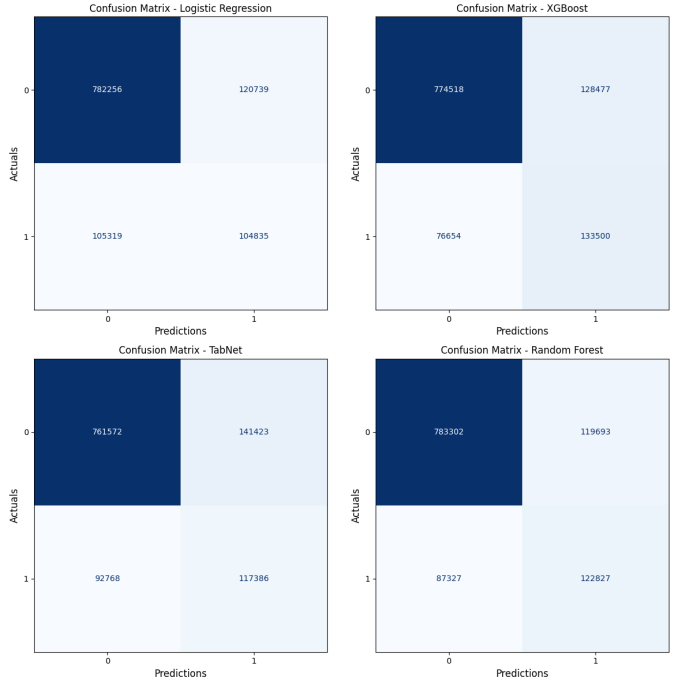

Figure 19: Four confusion matrices of the tuned models on ROS+RUS (hybrid) resampled data

APPENDIX E

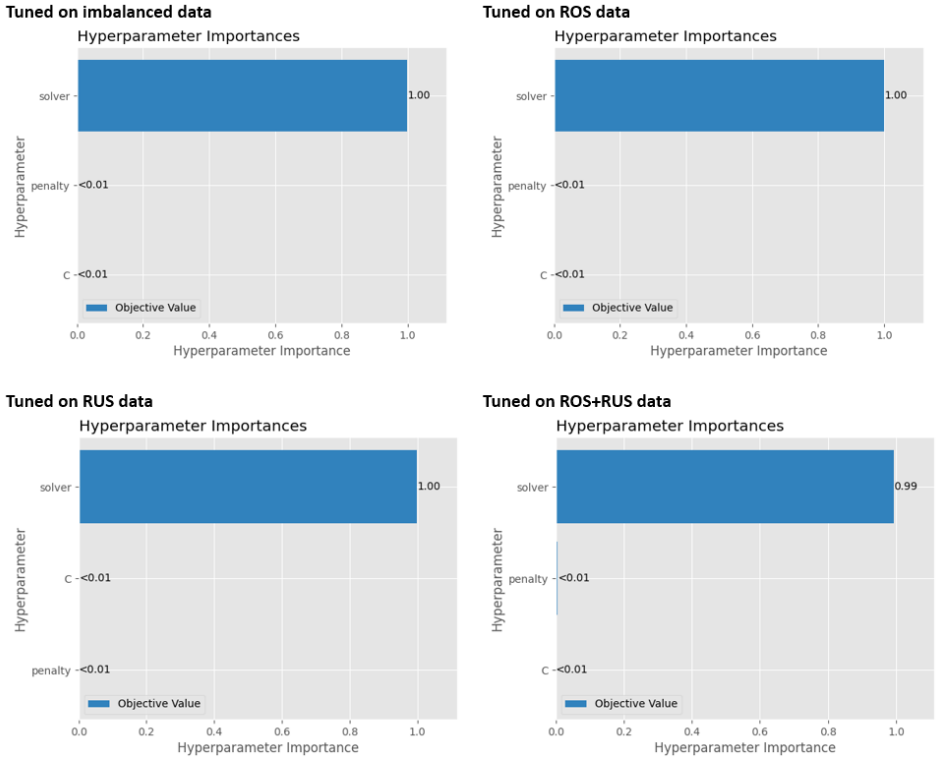

Figure 20: Optuna Feature Importance on Logistic Regression for each of the resampling strategies (including the imbalanced data)

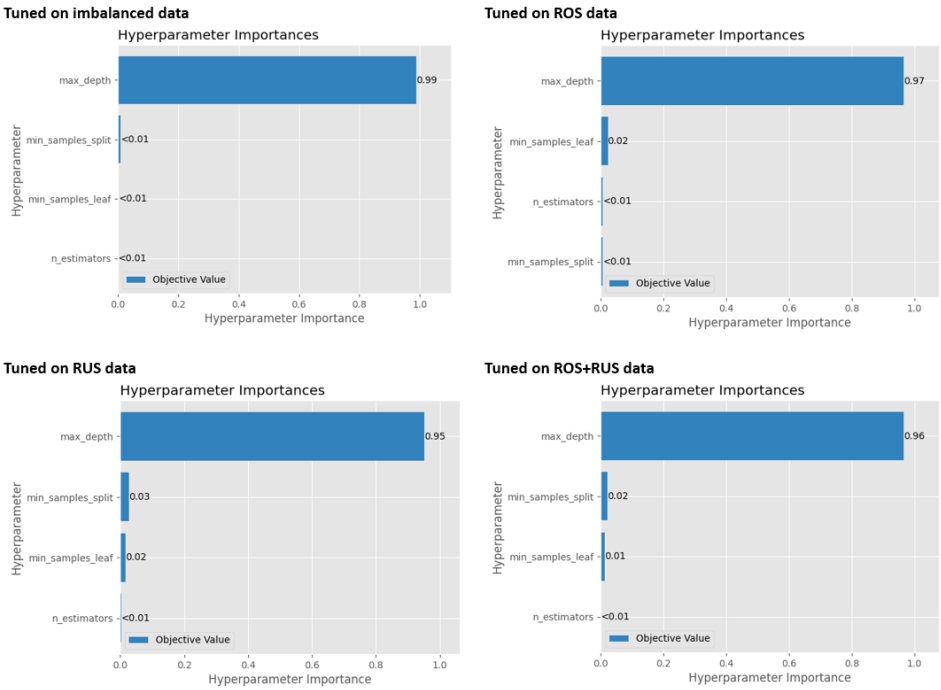

Figure 21: Optuna Feature Importance on Random Forests for each of the resampling strategies (including the imbalanced data)

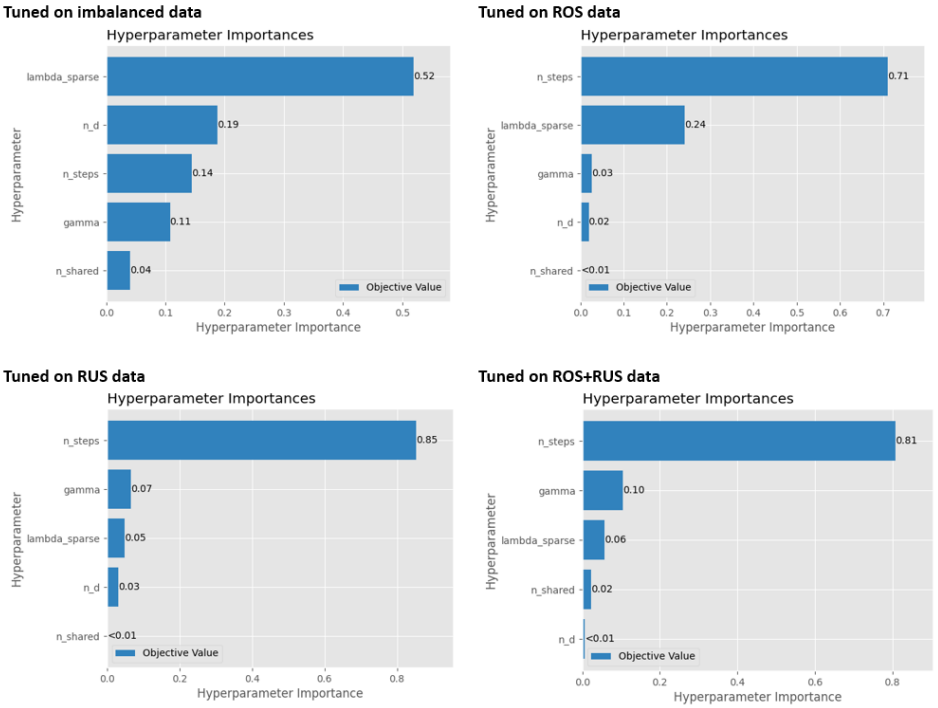

Figure 22: Optuna Feature Importance on TabNet for each of the resampling strategies (including the imbalanced data)

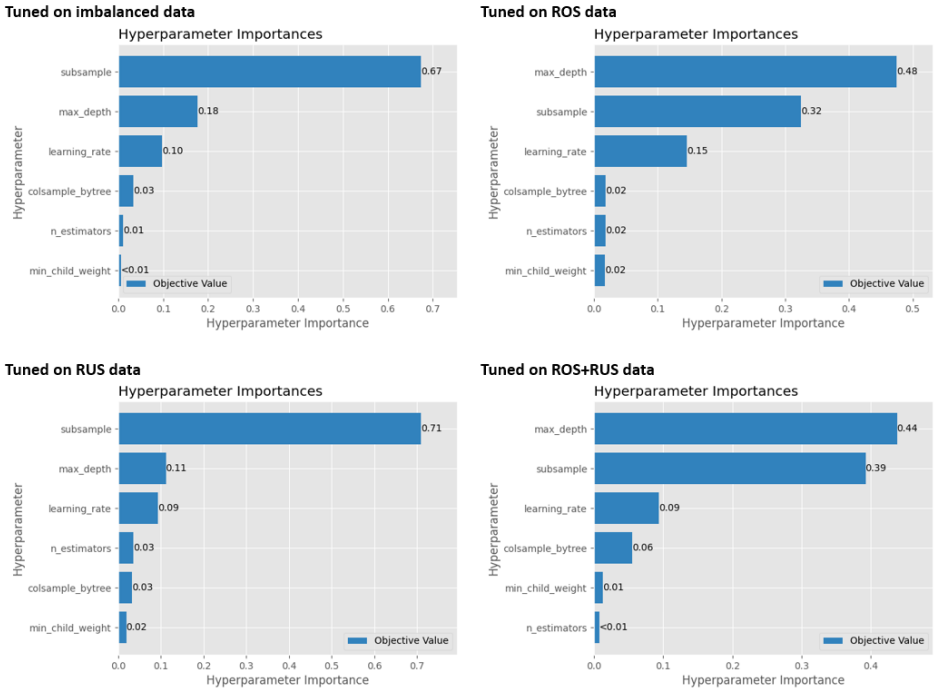

Figure 23: Optuna Feature Importance on XGBoost for each of the resampling strategies (including the imbalanced data)

APPENDIX F

This Appendix further describes the distribution analysis of features during EDA. Firstly, starting with the date related distributions (see Figure 24) such as: months, days of month and days of week. In general, we can conclude that this distribution is rather equal. Looking at day of the week it seems that Saturday is less frequent of an occurrence, this could be related to the dropping of cancellations and their being more cancellations on Saturday.

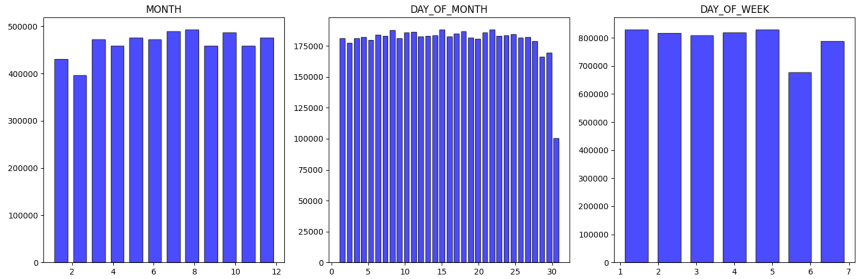

Figure 24: Distribution of months, days per month and days per week (Monday=1)

Next, we look at the distribution of the arrival and departure times (see Figure 25). That is noticeable here is that there are fewer to none airplanes leaving from 00:00 to 05:00, this can be explained by the fact that most airports are closed during that time or no flights due to noise regulations. Overall, the distribution seems quite equal.

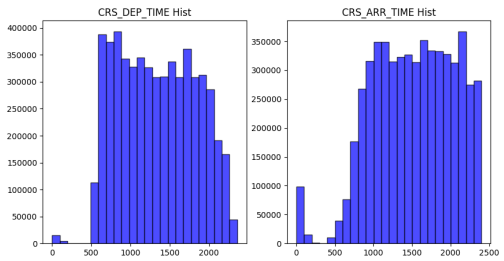

Figure 25: Planned Departure and Arrival times in military time format

Furthermore, when looking at Figure 26, we see that there are more flights making shorter trips in the U.S. then longer distances. The main conclusion being that these are just more popular. This distribution is slightly positively skewed. Moreover, a clear distinction can be made when comparing the amount of flight each carrier performs. This should be taken into account during the evaluation of RQ3.

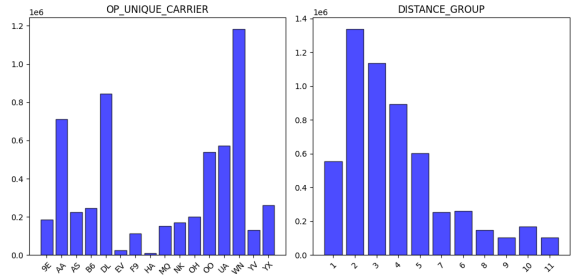

Figure 26: Distribution of flights per carrier and flights per distance group

Comparing individual planes we in Figure 27 we clearly see more modern planes being used then older models. This is rather normal as older models then to be less and less safe each year or needing more maintenance. Some years are more frequent then others, this can be explained by the fact that air carriers order their planes in batches. Comparing number of seats we see a few outliers. With the majority being around the 130-200 range.

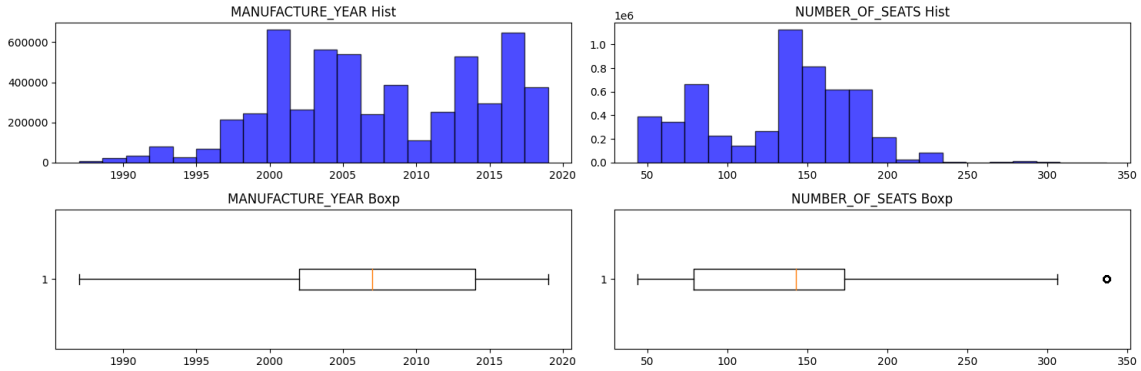

Figure 27: Distribution of manufacturing year and number of seats per plane, based on tailnumber

When comparing origins and destinations in Figure 28, we can see more destinations then origins. It could be that smaller airports don't report their departures to the BTS. If larger airports would be reporting more it would make sense that the destinations are more diverse.

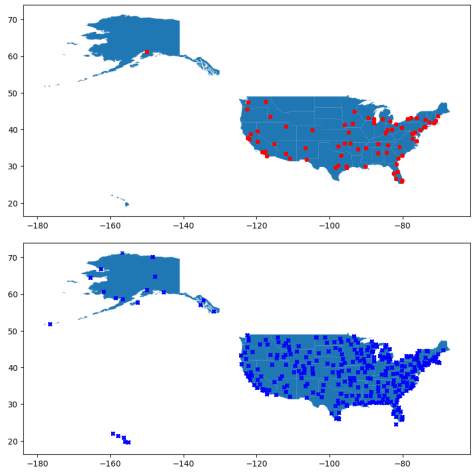

Figure 28: Top: Origin airport locations, Bottom: Destination airport locations

The demographic related distributions for the origin cities in Figure 29 do have a skewed distributions. Mainly a slight skewness to the positive side with some outliers. Because these outliers can be seen a s meaningful they will not be removed as this could impact other features.

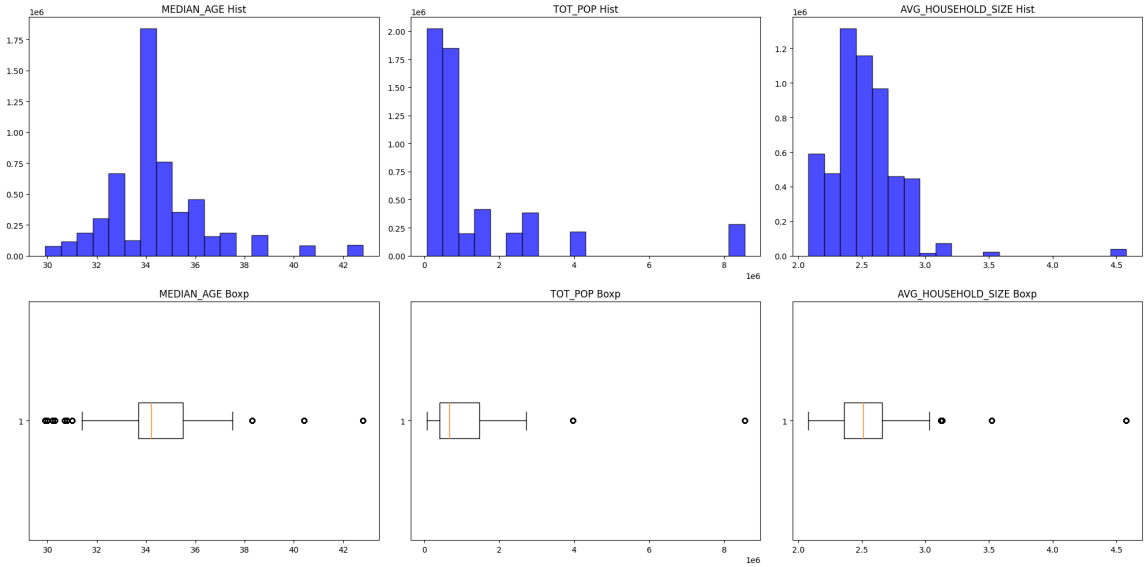

Figure 29: Distribution of manufacturing year and number of seats per plane, based on tailnumber

Lastly, the weather related features and their distributions (see Figure 30). All these show to have high skewness, kurtosis and outliers. With precipitation and snow standing out as having many outliers, even after pre-processing. Again this can be explained, as it is less common for it to

be raining a lot or snowing. These datapoints again could be insightful as they could impact delays. Thus they will not be removed but instead scaled. The same does for temperature and wind, while they have a more normal distribution, they still have many outliers. Again, these could play an important role in the predictability of delays and will not be removed.

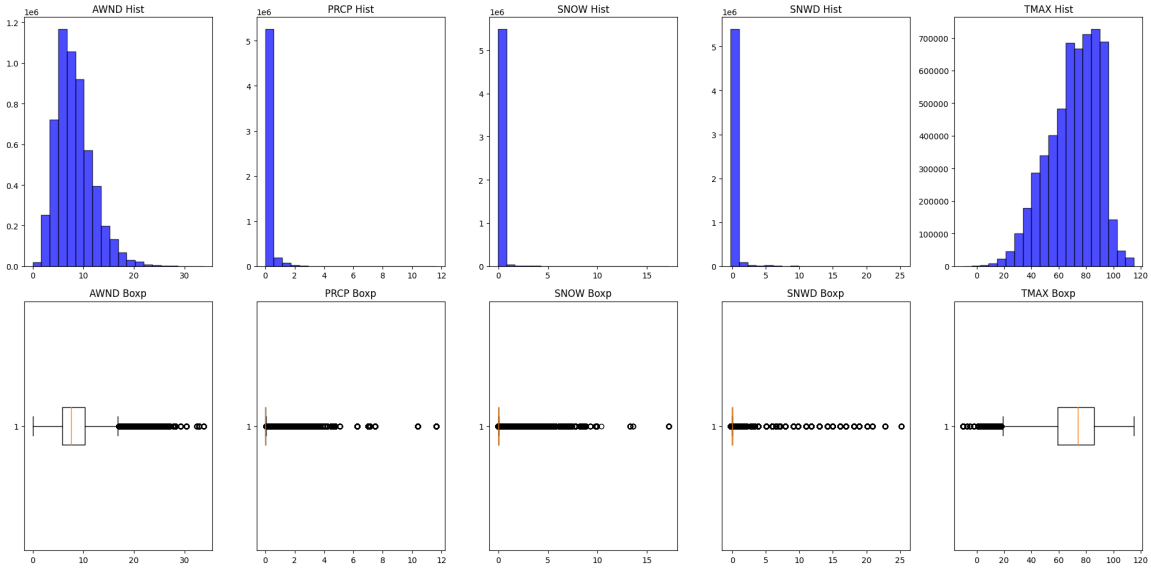

Figure 30: Distribution of weather related features