Application of Interpretable Machine Learning
in Flight Delay Detection

by

AFROZA HOSSAIN

MASTERS THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Systems at
The University of Texas at Arlington
May, 2021

Arlington, Texas

Supervising Committee:

Dr. Manjeri K Raja, Supervising Professor
Dr. Riyaz Sikora
Dr. Jayarajan Samuel

ABSTRACT

Application of Interpretable Machine Learning in Flight Delay Detection

Afroza Hossain (Masters of Science in Information System)

Supervising Professor: Dr. Manjeri K Raja

Precise flight delay prediction is vital for the airline industries and passengers. This thesis focuses on applying several machine learning and auto-ML techniques to predict flight delays. A flight delay is said to occur when an airline lands or takes off later than its scheduled arrival or departure time, respectively. Conventionally, if a flight's departure time or arrival time is greater than 15 minutes than its scheduled departure and arrival times respectively, then it is considered that there is a departure or arrival delay with respect to the corresponding airports. Notable reasons for commercially scheduled flights to be delayed are adverse weather conditions, air traffic congestion, a late reaching aircraft to be used for the flight from a previous flight, maintenance, and security issues. In this research study, a python-based model will be developed for a specific Airline and an Airport from already established models that are available in literature and were implemented in flight delay predictions. Once that is completed, the same model will be used for a different Airline at the same Airport. Later, the model will be implemented for several other Airports to check the adaptability of the models. In this process, there will be an attempt to enhance the existing models by carefully selecting the dataset and features. In the final stage, the results will be compared with the Microsoft Azure Machine Learning Studio, the best model will be

deployed using Auto-ML and the existing interpretable machine learning package, LIME will be used to explore local prediction capability of the models. This study has been conducted with the hopes that alongside other increasing numbers of studies in this subject matter, it will contribute to improving on-time performances of flights to benefit airline customers, airline personnel, and airport authorities.

ACKNOWLEDGEMENTS

Foremost, I would like to thank Prof. Manjeri K Raja, my advisor and Professor of Information Systems and Operations Management at the University of Texas at Arlington, for his advises and guidance, he gave me an interesting insight on the possible applications. Besides my advisor, I would like to thank the rest of my thesis committee member: Professor Riyaz Sikora, Professor Jayarajan Samuel for their inspiration. I offer my sincere appreciation for the learning opportunity provided by University of Texas at Arlington and my committee.

DEDICATION

I dedicate this thesis to my encouraging husband who has been a constant source of support and inspiration during the challenges of school and life. This effort is also dedicated to my parents and my sister, as their unconditional love and prayers made it possible to receive such an honor.

TABLE OF CONTENTS

ABSTRACT 1

ACKNOWLEDGEMENTS 4

DEDICATION 5

Chapter 1 11

INTRODUCTION 11

Chapter 2 13

LITERATURE REVIEW 13

APPLICATION OF MACHINE LEARNING IN FLIGHT DELAY PREDICTION 13

MACHINE LEARNING WORKFLOW 14

Dealing with Imbalanced data 19

Interpretable Machine Learning: 26

Chapter 3 29

DATA GATHERING PREPROCESSING AND EXPLORATION 29

EDA: Exploratory Data Analysis for American Airlines 37

EDA: Exploratory Data Analysis for Southwest Airlines 50

Chapter 4 53

RESULT AND DISCUSSION 53

MODELS using Python for American Airlines 53

MODELS using Python for Southwest Airlines 57

Microsoft Azure Studio 59

Interpretability of the models prediction using LIME: 62

All Model Comparison for LIME for a specific Flight from test data set 62

Chapter 5 74

CONCLUSION 74

REFERENCE 76

APPENDIX 77

LIST OF FIGURES (or Illustrations)

Figure 1: A typical process of Air Transportation System (Sternberg et al., 2017) 11

Figure 2: Flow Chart of the Research Plan for Interpretable Flight Delay Prediction 12

Figure 3:Publication in last two decades in Flight Delay Prediction Statistical Analysis, Probabilistic Models, Network Representation, Operational Research, Machine Learning (Sternberg et al., 2017)... 14

Figure 4:Schematic of generic Machine Learning workflow 15

Figure 5: Different sources of data for machine learning models 16

Figure 6:Process of splitting the data set 18

Figure 7:Example of imbalanced dataset.. 19

Figure 8: Oversampling example, creating balance.. 20

Figure 9:SMOTE (Bank Data: SMOTE. This Will Be a Short Post before We… | by Zaki Jefferson | Analytics Vidhya | Medium, n.d.) 20

Figure 10: Undersampling technique example, creating balance 21

Figure 11:Adaboost Classifier iteration approach 23

Figure 12: Multilayer perceptron-NN... 25

Figure 13: Importance of interpretable ML 27

Figure 14: schematic of how LIME works 27

Figure 15: Steps that LIME uses for the interpretation... 28

Figure 16:Heat Map 37

Figure 17: Total number of AA flights that arrived at DFW 38

Figure 18: Total Delayed AA Flights by origin airport based in arrival delay status.................. 39

Figure 19:Comparison pie char between arrival and delay status 40

Figure 20:Percentage of delayed arrival flights of airport 40

Figure 21: Percentage of departure delayed flights by airport... 41

Figure 22:Total delays in minutes by airport.. 41

Figure 23:Average arrival delay by airport... 42

Figure 24:Number of delays flight per month 43

Figure 25: Number of Delayed flights per day of week 43

Figure 26: Number of delayed flights per day of the month.. 44

Figure 27:Departure vs arrival delay 45

Figure 28:Total delayed AA flights by origin airport based on carrier delay.............................. 46
Figure 29:Total delayed AA flights by origin airport based on weather delay 47
Figure 30:Total delayed AA flights by origin airport based on NAS delay 47
Figure 31:Total delayed AA flights by origin airport based on security delay 48
Figure 32: Total delayed AA flights by origin airport based on late air-craft delay 49
Figure 33:Percentage of flights delays based on carrier, weather, NAS, security, late aircraft ... 50
Figure 34: Total number of Southwest flights arrived at DAL... 50
Figure 35: Total delayed SW flights based on arrival delay... 51
Figure 36:Comparison between departure and arrival delay 52
Figure 37: Total delayed SW flights by origin based on cause of delay 52
Figure 40:Evaluation Metric for random forest 55
Figure 42:Evaluation Metric for Adaboost 55
Figure 40: Evaluation Metric for XGBoost 56
Figure 44: Evaluation Metric for MLP-NN 57
Figure 45:Evaluation Metric for Random Forest 57
Figure 47:Evaluation Metric for Adaboost 57
Figure 48:Evaluation Metric for xgboost.. 58
Figure 49: Evaluation Metric for MLP- NN.. 58
Figure 50: Visualization from Azure 60
Figure 51: Result for Two-Class Decision Forest... 61
Figure 52:Result for Azure Auto-ML 62
Figure 53: Case 1 result using Random Forest 63
Figure 54: Case 1 result using Adaboost 63
Figure 55: Case 1 result using Gradient Boosting 64
Figure 56 : Case 1 result using xgboost... 64
Figure 57 : Case 2 result using Random Forest 65
Figure 58 : Case 2 result using Adaboost 65
Figure 59 : Case 2 result using Gradient Boosting 66
Figure 60: Case 2 result using XGBoosting 66
Figure 61 Random Forest model was run with Flight Number 71
Figure 62 Random Forest model was run without Flight Number 71

Figure 63: Case 1: Delay flight based on customer centric Interpretable prediction.................... 72

Figure 64 Case 2: Flight arrive on time based on customer centric Interpretable prediction....... 73

LIST OF TABLES

Table 1:Features with description 30
Table 2: Performance metrics for AA 59
Table 3:: Performance metrics for Southwest 59

Chapter 1

INTRODUCTION

An arrival flight delay is said to occur when an airline lands later than its scheduled arrival time. Notable reasons for commercially scheduled flights to delay are adverse weather conditions, air traffic congestion, late reaching aircraft to be used for the flight from previous flight, maintenance and security issues.

Flight delays are relatively common in both domestic and international flights. Based on the statistics of the Bureau of Transportation, 18% (average of last 10 years)((*Bureau of Transportation Statistics*, n.d.) of US domestic flights arrive more than 15 minutes late. According to Baik et al., these delays are not only a cause of frustration for the passengers, Airline, and airport authorities but also play an important role in financial loss for all parties (Baik et al., 2010)). Some of the flights are more frequently delayed than others. The continuous technological advancement in data storage enables the storage of a massive amount of data and computational power leads to the development of data analytics. Different government agencies, airport authorities, and Airline companies are collecting significant amounts of data and analyzing these datasets to aid in gaining knowledge about the delays. A robust flight delay prediction model with proper explanation of the delay is not only an interest of travelers but also of Airlines and Airport Authorities.

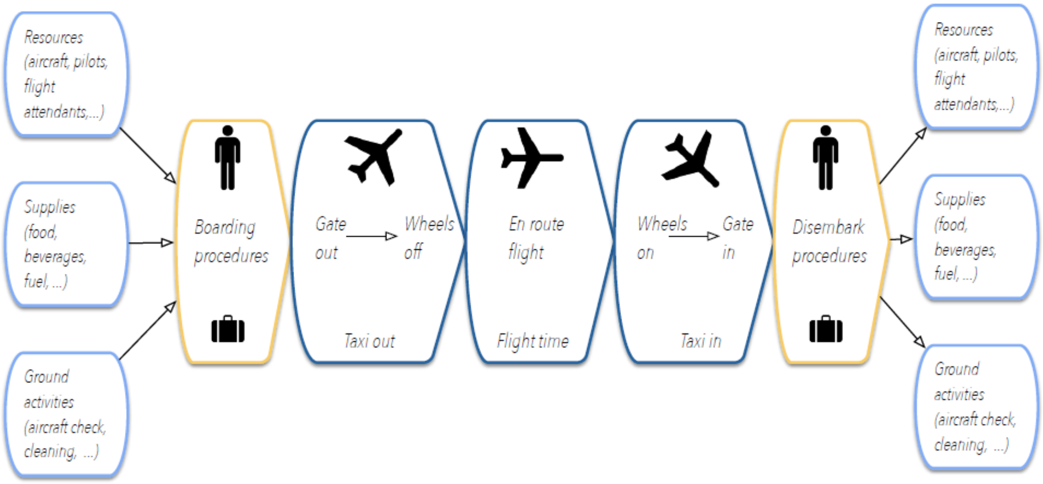

Resources (aircraft, pilots, flight attendants,...)
Supplies (food, beverages, fuel, ...)
Ground activities (aircraft check, cleaning, ...)
Boarding procedures
Gate out
Wheels off
Taxi out
En route flight
Flight time
Wheels on
Gate in
Taxi in
Disembark procedures
Resources (aircraft, pilots, flight attendants,...)
Supplies (food, beverages, fuel, ...)
Ground activities (aircraft check, cleaning, ...)

Figure 1: A typical process of Air Transportation System (Sternberg et al., 2017)

PROBLEM STATEMENT & RESEARCH PLAN

Air traffic is a very complex transportation system and the reason for the delay can occur at different stages of the process. Figure 1 is a schematic of this complex process.

In this research study, a python-based model will be developed for a specific Airline and one of their operating or hub Airport from already established models that are available in the literature and were implemented in flight delay predictions. These data-driven methods will only consider historic observations and will be using several years of records (largest public dataset of flight delay) from the Bureau of Transportation Statistics (BTS) of the United States Department of Transportation of US domestic flight delays. The model will not take into account short-term effects, such as current weather or traffic situation. The same model will then be used for a different Airline and it's one of the operating or hub Airports. In this process, there will be an attempt to enhance the existing models by carefully selecting the dataset and features. In the final stage, the results will be compared with Microsoft Azure Machine Learning Studio and also with Azure Auto-ML, and then the interpretation of prediction of delay will be made based on an interpretable machine learning package, Local Interpretable Model-Agnostic Explanations (LIME). This study is being conducted with the hopes that it will contribute to improving on-time performances of flights for the benefit of the airline customers, airline personnel, and airport authorities. Figure 2, which can be found below, represents the Research Plan for Flight Prediction Model and the interpretation of the prediction.

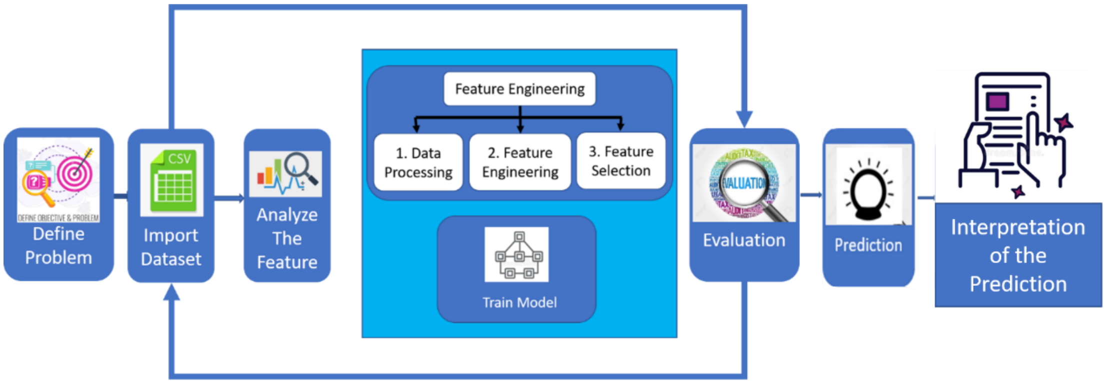


Figure 2: Flow Chart of the Research Plan for Interpretable Flight Delay Prediction

Chapter 2

LITERATURE REVIEW

APPLICATION OF MACHINE LEARNING IN FLIGHT DELAY PREDICTION

Flight delays are an important subject in literature due to their economic and environmental impacts. They may increase costs to customers and operational costs to airlines. Apart from outcomes directly related to passengers, delay prediction is vital during the administrative process for all parties involved in the air transportation system, this has been explored in Annual U.S. Impact of Flight Delays by FAA (Apo-, 2016). Approximately $23 billion loss was reported due to delay of flights in the U.S as per the yearly reports of FAA, these include both domestic and international flights. In the last few decades, many attempts of flight delay predictions have been made by researchers based on Machine Learning, Deep Learning, and Big Data approach (Figure 2). A regression model using Gradient Boosting Regressor for predicting both Flight Departure and Arrival Delays was explored and analyzed by Manna et al. and Yazdi et al.(Yazdi et al., 2020a)(Manna et al., 2018). Applied Supervised Machine Learning Algorithms like decision trees, random forests, AdaBoost, and k-Nearest Neighbours for predicting weather influenced flight delay were reported by Choi et al.(Choi et al., 2016). Rebollo et al.(Rebollo & Balakrishnan, 2014) applied Random Forest on an air traffic network framework for predicting flight departure delays in the future. Different machine learning methods such as decision trees were analyzed by Kuhn et al. and Dothang et al. (Truong et al., 2018) (Kuhn & Jamadagni, 2017), random forest models were used in Predictive Modeling of Aircraft Flight Delay by Kalliguddi et al. (Kalliguddi et al., 2017), naïve Bayes model was used in Prediction Analysis of Flight Cancellation Based on Spark, bagging classifier, extra trees classifier, gradient boosting methods were used by Yanying et al.(Yanying et al., 2019). A recent article by Yazdi et al. (Yazdi et al., 2020b) proposes a model for predicting flight delay based on Deep Learning (DL).

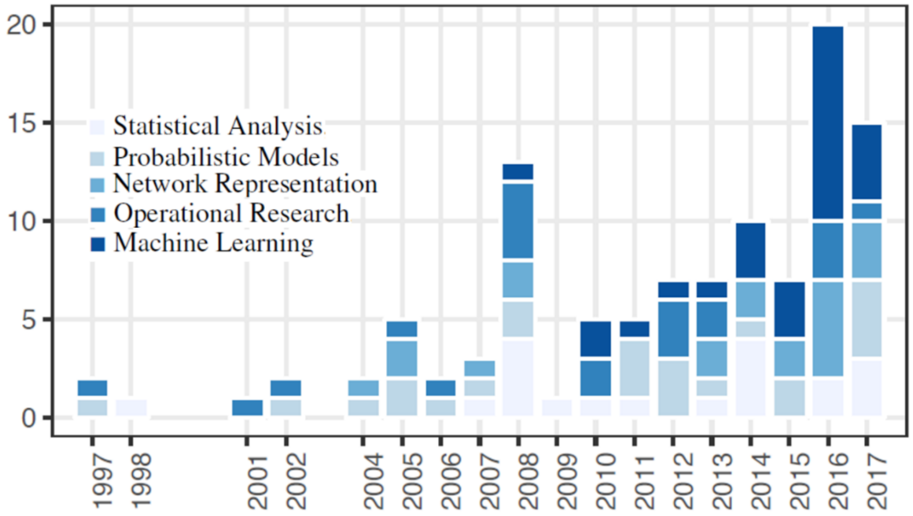


Figure 3:Publication in last two decades in Flight Delay Prediction Statistical Analysis, Probabilistic Models, Network Representation, Operational Research, Machine Learning (Sternberg et al., 2017)

According to Teja et al, (Teja, n.d.) flight delays do not only have an economic impact but also have an environmental impact, this was determined using machine learning algorithms like XGBoost regression and Linear regression Techniques. Another model which was used to predict flight delay by Sina (Gui et al., 2020) tries to reduce delays to gain the loyalty of their customers using artificial neural network (ANN) techniques.

MACHINE LEARNING WORKFLOW

Machine Learning (ML) models are used to learn from data without being explicitly programmed. ML models are code that has been trained to recognize several types of patterns in the data and make a prediction based on that. Machine learning techniques are useful for solving experiments efficiently and effectively. A great amount of data is loaded into a computer program and a model is chosen to fit the data, which allows the program, without any help to make predictions, based on the trained data. Predictive models are only as good as the data from which they are built, thus using valid and relevant data helps with high-performing models. Here, the analogy of garbage-in garbage-out takes into effect which means that if a model is fed garbage, that is exactly what it will return, in other words, the trained model will provide invalid predictions. Based on Figure 3, the workflow of Machine learning includes all the steps required to build the proper machine learning model from scratch.

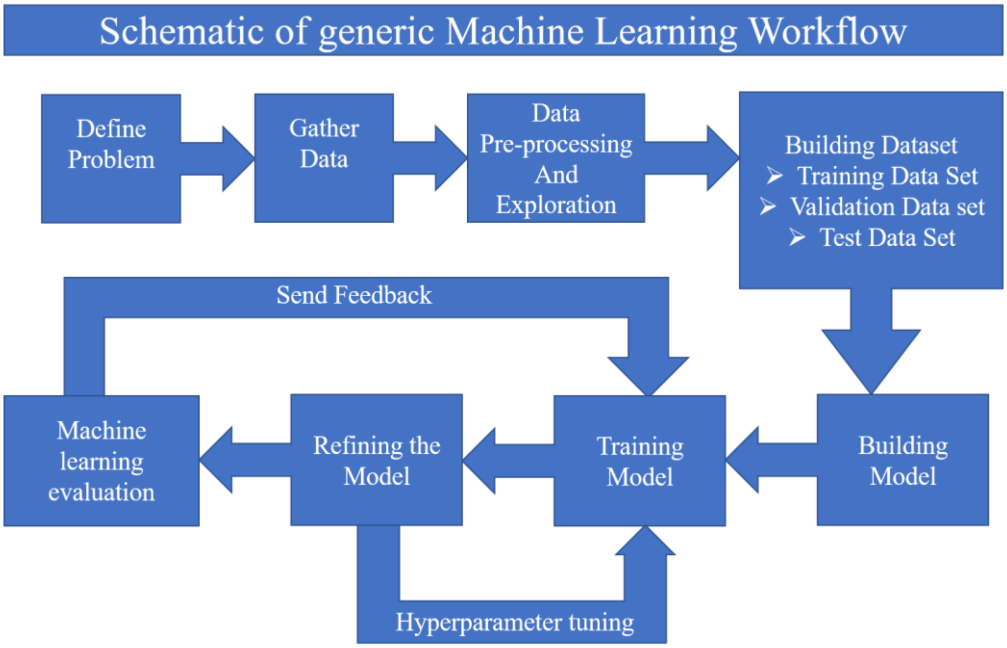


Figure 4:Schematic of generic Machine Learning workflow

Define Problem: This is the first step of machine learning. It is important to identify what exactly one expects as output from a model. This might involve in having some assumptions which mostly come from the domain knowledge.

Gather Data: Based on one problem statement, one must gather an appropriate data set. Quality of data dictates the accuracy of the model, so it is a very important stage of ML workflow. This can be a tedious process because getting the right dataset in the right format can be challenging. It has to correlate with the outcomes which are being predicted, accumulating data, requires a clear understanding of domain knowledge, and proper engagement in sampling from a large database to capture records to be used in an analysis. Figure (below) is a depiction of different sources of data for machine learning models.

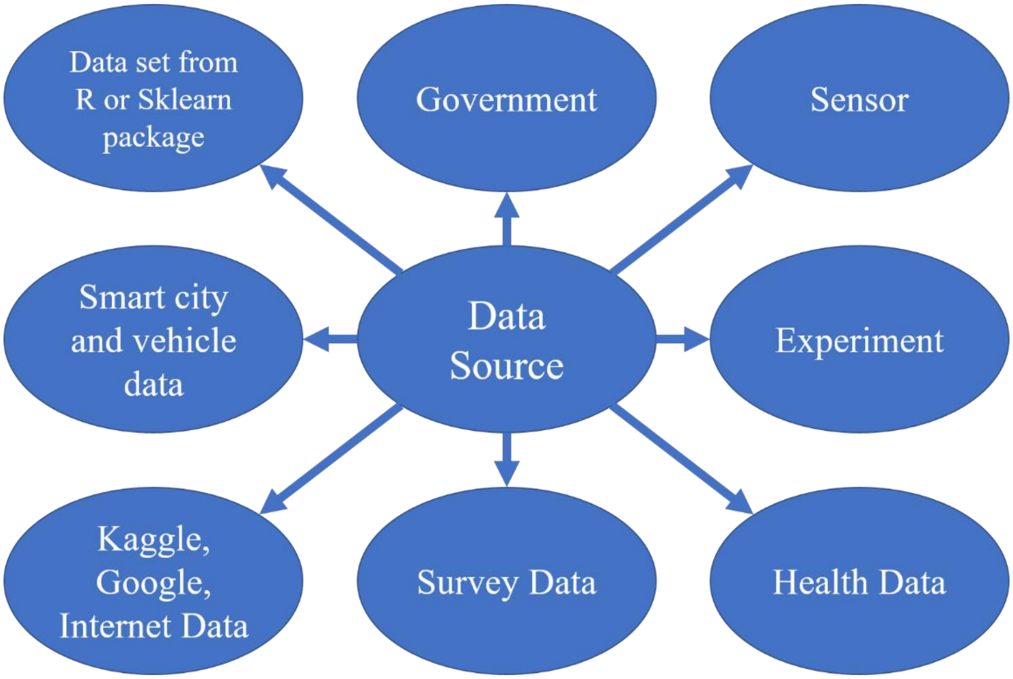

Data set from R or Sklearn package
Government
Sensor
Smart city and vehicle data
Data Source
Experiment
Kaggle, Google, Internet Data
Survey Data
Health Data

Figure 5: Different sources of data for machine learning models

Data Pre-processing: Data preprocessing is the process of cleaning data and preparing it to be used to train the model. Most scientists believe data cleaning and formatting can be considered the most challenging part of any project (Shmueli et al., 2017). In the real world, data is incomplete, inconsistent, and inaccurate which means that there are errors and outliers present in the data which causes there to be a lack in patterns and trends. According to (Shmueli et al., 2017) data preprocessing enhances the quality of data to stimulate the extraction of meaningful insights. Some of the key steps of data preprocessing are

a) gathering the dataset: In this case, data was gathered from the Bureau of Transportation Statistics for 2016-2019, using only American Airlines (AA) and Southwest (WN), limited to the US domestic flights and a couple of airports.

b) Importing all the required libraries: importing libraries and dependencies, into the Python environment will make tasks easier, as it has built-in functions and models that can be used instead of doing that ourselves. For example, some of the libraries that will be used are pandas, which is used for data cleaning and analysis, NumPy which is a library that is mostly used for, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays, matplotlib which is used for analyzing and visualizing charts or graphs, as well as several other libraries which will be used throughout the project.

c) Importing the dataset: When running python programs, datasets are required for data analysis. Python has several modules for importing external data. For example, the method used in this project is by importing the CSV to enable us to read each row in the file using a comma as a delimiter, which is best described in (*Importing Data in Python*, n.d.)

d) The next step in ML is feature engineering, it is the process of using domain knowledge to extract features from raw data. These features can be used to improve the performance of machine learning algorithms. The following list of techniques exist in feature engineering: Imputation, Handling Outliers, Binning, Log Transform, One-Hot Encoding, Grouping Operations, Scaling, some of which will be used in this thesis.

i. Data cleaning or identifying and handling the missing values: according to (Shmueli et al., 2017) from a statistics point of view, it is important to understand different types of missing data. The type of missing data will help with filling in the data or discarding them. In python, *isnull* and *notnull* can also be used to summarize missing values, this will be discussed in greater detail further in this paper.

ii. Encoding the categorical data: According to (Shmueli et al., 2017) data scientists spend 80% of their time cleaning and preparing data, in this process converting categorical data is an inevitable activity. It helps improve the model quality and provides better feature engineering. For example, in some of the feature binary variables containing either 0 or 1, where 0 represents the absence, and 1 represents the presence.

iii. Feature Scaling is also sometimes involved in this step of ML workflow. It helps to normalize the data through standardization, which involves rescaling the properties of a standard normal distribution with a mean of zero and a standard deviation of one. Normalization is a scaling technique in which values are rescaled, hence varying between zero and one, it is also known as Min-Max scaling. **Error! Reference source not found.** is the formula for normalization where Xmax and Xmin signify the maximum and the minimum values of the feature, respectively.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

iv. Another type of scaling often used is Standardization where the values are focused around the mean with a component standard deviation. **Error! Reference source not found.** is the formula for standardization, where μ represents the mean of the feature values and σ represents the standard deviation of the feature values.

$$X' = \frac{X - \mu}{\sigma}$$

Once preprocessing has been complete, data exploration is the first step in data analysis where the use of data visualization and statistical techniques describes the nature of the dataset. Data exploration helps us visually explore and categorize relationships between different features, structures, and the presence of outliers. It also helps us understand the patterns and trends present in the raw data.

Building Data set: Based on the data, and the purpose of the problem, one has to determine the Machine Learning task to be performed amongst the following tasks: classification, prediction, clustering, and partitioning the data accordingly. For example, if the problem is a supervised (prediction), the dataset is divided into three parts: training, validation, and test datasets. Many Machine Learning techniques such as regression, neural nets, decision trees, etc, can be used in this iterative process. Below figure is an example of the process of splitting the data set.

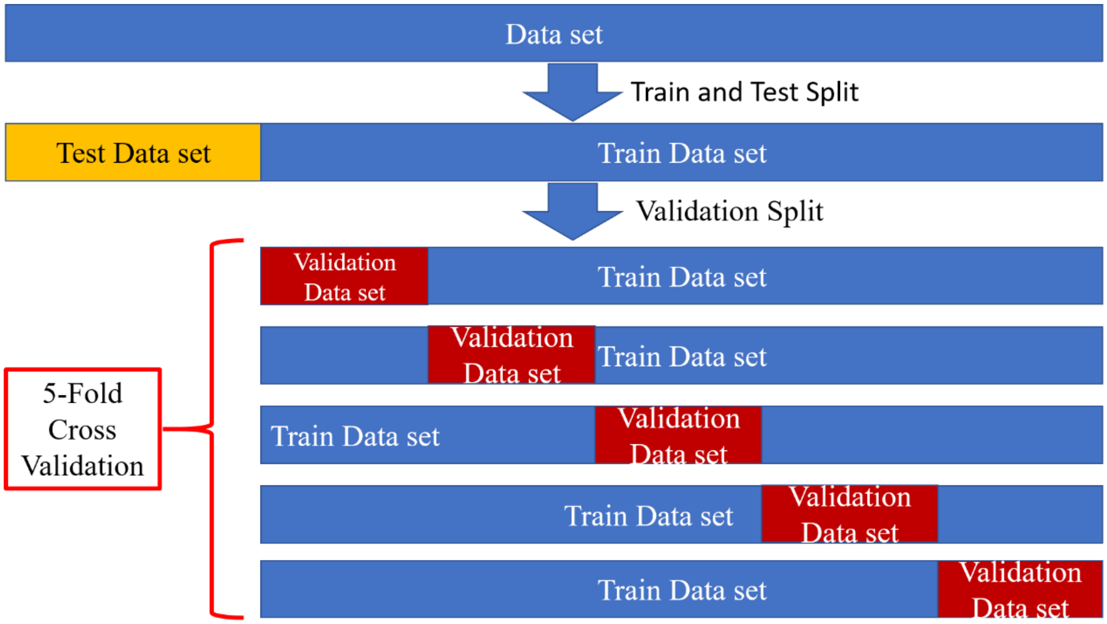


Figure 6:Process of splitting the data set

Dealing with Imbalanced data

As most of the machine learning algorithms are designed to maximize accuracy and reduce error, it works best when the number of samples in each class are about equal but in real data set this is very rare.

Class imbalance is a problem that arises in machine learning classification problems. A classification problem in machine learning in which a target must be predicted given some input. There is a great chance that the distribution values may be different and due to this difference in the class, the algorithms may be biased towards the majority values and will not perform well on the minority values, as a result, the difference in class affects the outcome of the model.

For an example, if someone has a two-class problem (e.g., yes or no). If 10% data points are of the class of "Yes", 90% for the class of "No" class. The "No" class is the majority class and "Yes" class is the minority. Here "No" class to "Yes" class ratio is very high. This problem is referred to as a class imbalance.

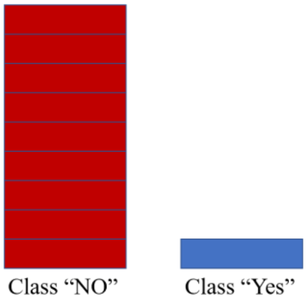


Figure 7:Example of imbalanced dataset

There are many techniques which are used in dealing with the imbalanced data. Below are some examples of those techniques.

Oversampling: This technique tries to increase the size of minority samples to create a balance.

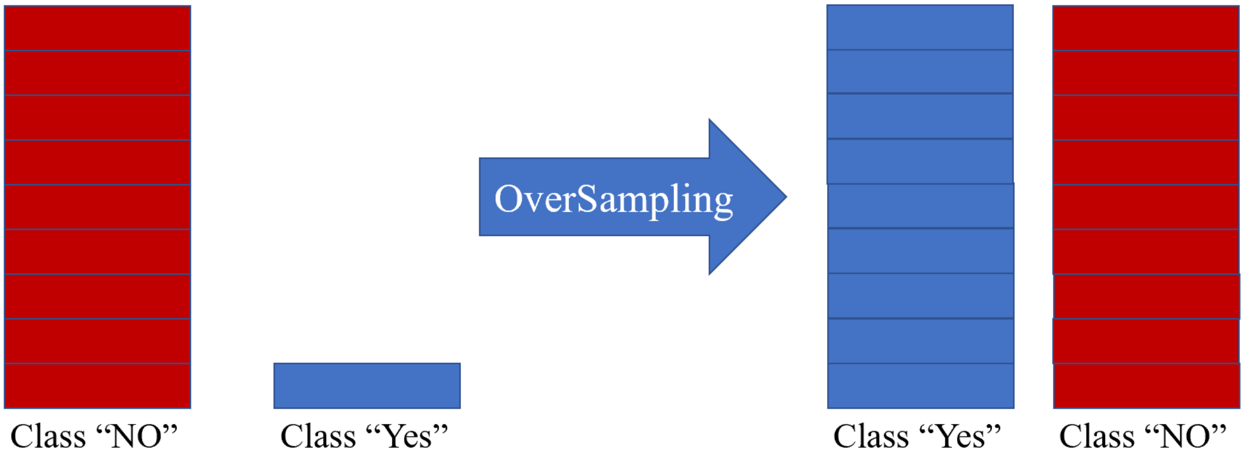


Figure 8: Oversampling example, creating balance

One of the popular oversampling method is Synthetic Minority Over-sampling Technique (SMOTE). By using a distance measure, SMOTE algorithm selects two or more similar instances to create synthetic samples from the minority class. below is a schematic of the algorithm.

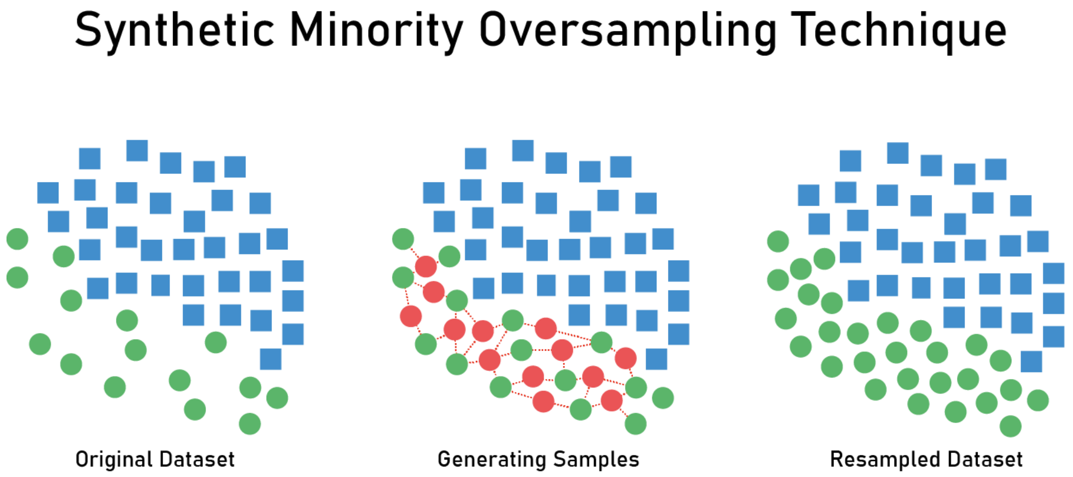


Figure 9:SMOTE (Bank Data: SMOTE. This Will Be a Short Post before We… | by Zaki Jefferson | Analytics Vidhya | Medium, n.d.)

Undersampling: Undersampling works in opposite way of oversampling, it aims to decrease the size of the majority class to balance the dataset.

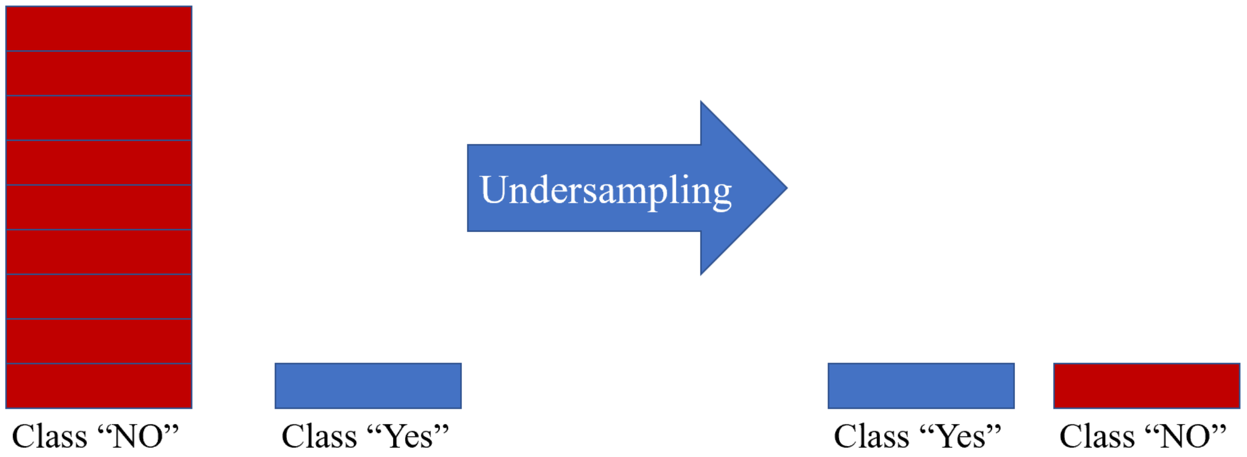


Figure 10: Undersampling technique example, creating balance

Penalize Model: One can penalize the model to pay more attention to the minority class by imposing an additional cost on the model for making classification mistakes on the minority class during training.

Assigning Weights for the Classes: As stated earlier, machine learning algorithms are not very useful with biased class data. When training algorithms, skewed distribution of the classes can be taken into consideration. This can be accomplished by giving different weights to both the majority and minority classes. The formula to calculate weights is **Error! Reference source not found.**:

$$wj=n\_samples / (n\_classes * n\_samplesj)$$

Where,

wj represents the weight for each class

n_samples represents the total number of samples

n_classes represents the total number of unique classes in the target

n_samples represents the total number of rows of the respective class

In order to achieve more accurate and meaningful results, weights were distributed among the classes.

Building Model: Before diving into models, understanding algorithms is important, Machine learning algorithms can be divided into 3 broad categories:

- Supervised learning
- Unsupervised learning, and

- Semi-supervised learning.

Supervised learning is the function that maps an input to an output based on example input-output pairs. Unsupervised learning is a type of algorithm that learns patterns from untagged data. According to (Truong et al., 2018), semi-supervised learning is an approach that combines a small amount of labeled data with a large amount of unlabeled data during training. As can be interpreted from the name, semi-supervised learning falls between unsupervised learning and supervised learning. Machine learning algorithms build a model based on sample data, identified as the "training data", to make predictions. There are many models in Machine Learning but the following four models: Random Forest, AdaBoost, XgBoost, Neural network will be looked into.

Random forest: Random forest is one of the most used algorithms, it can be used for both classification and regression tasks, classification will be used in the case of this thesis. To use this algorithm in Python, the RandomForest Classifier and BaggingClassifier libraries are required to be imported. It is the supervised, ensemble learning, usually trained with the “bagging” method which helps with the overall result. The Random Forest algorithm will be used because it is very easy to measure the importance of each feature in the prediction. The hyperparameters are used to increase the predictive power of the model or to make the model quicker. Some common hyperparameters for increasing the predictive power are the n_estimators which is the number of trees the algorithm builds. In general, a higher number of trees increases the performance and makes the predictions more stable. Another important hyperparameter is max_features, which is the maximum number of features random forest considers when splitting a node. The last hyperparameter is min_sample_leaf which helps determine the minimum number of leafs required to split an internal node. Although the Random Forest algorithm is versatile, it is slow in creating predictions once the model is made. The Random Forest algorithm uses the Gini Index which is a measure for classification type problems. For the purpose of this thesis, the function being used takes in the parameters gini which measures the quality of a split and the maximum depth of the tree. If the maximum depth is denoted to none, then the nodes are expanded until all leaves contain less than min_samples_split samples.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

The formula **Error! Reference source not found.** uses probability to determine the Gini of each branch on a node; deciding which branch is more likely to occur. Here, *P_i* represents the relative frequency of the class in the dataset and *c* represents the number of classes. This can also be done using entropy. The formula for entropy is shown below:

$$Entropy = \sum_{i=1}^{C} - p_i * \log_2(p_i)$$

Entropy manages the probability of a certain outcome for making a decision (the equation above). This algorithm can be extremely useful with different types of data sets. It is easy to use, fast to train and finds an accurate representation.

AdaBoost: The next model which is used in this thesis is AdaBoost, short for "Adaptive Boosting. To use this algorithm in Python, the AdaBoostClassifier library needs to be imported. It focuses on classification problems and uses an ensemble learning method. AdaBoost uses an iterative approach to learn from the mistakes of weak classifiers, and turns them into strong ones as shown below.

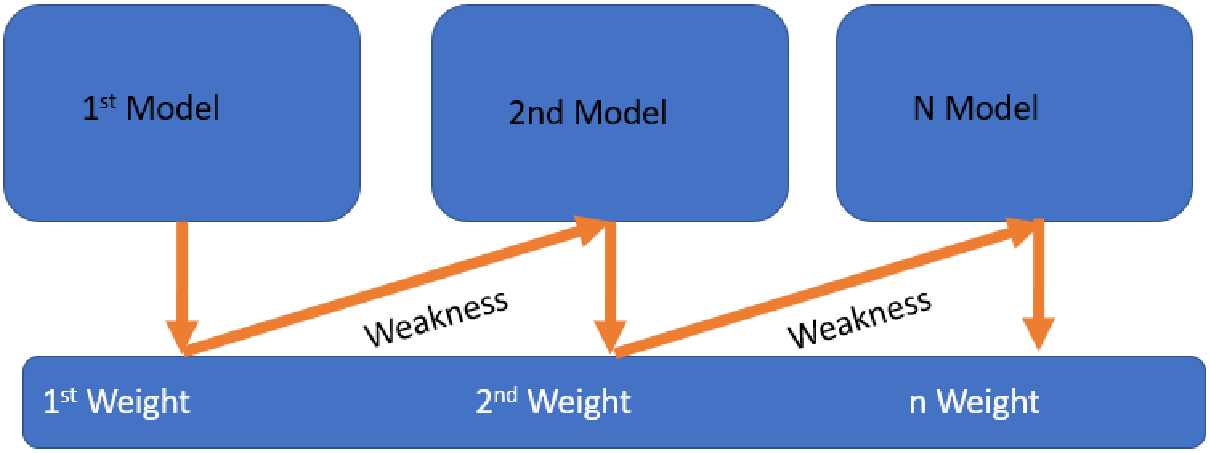


Figure 11:Adaboost Classifier iteration approach

The way the model works is, it makes n number of decision trees during the training, as the first decision tree is made, and the record which is incorrectly classified is given more priority. Two hyperparameters that are mostly used are the number of estimators (n_estimators) and the learning rate. Only those incorrect records are sent as input for the next model and the process will continue. The final equation for classification can be represented **Error! Reference source not found.**:

$$F(x) = sign\left(\sum_{m=1}^{M} \theta_m f_m(x)\right)$$

where f_m stands for the m’th weak classifier and θ_m is the corresponding weight. It represents the weighted combination of M weak classifiers.

XGBoost: Another algorithm which will be looked into is XGBoost, it is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. It helps in prediction problems involving unstructured data. This algorithm has many advantages such as Regularization which also helps to reduce overfitting. Overfitting occurs when the model tries to contain all the data points in the provided dataset, which reduces the accuracy of the model. The overfitted model results in low bias and high variance. XGBoost has an in-built procedure to handle missing values on each node, which allows users to run cross-validation at each iteration of the boosting process. The final equation for XGBoost can be represented as **Error! Reference source not found.**:

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right]$$

Multilayer Perceptron (MLP): The last algorithm/model which will be used in this project is Multilayer Perceptron (MLP). MLPs are suitable for classification prediction problems. It utilizes a supervised learning technique called backpropagation. It is an algorithm that calculates a complicated gradient. In the Multilayer perceptron, there are combinations of neurons. For instance, in a three-layer network, the first layer is the input layer, the middle layer is the hidden layer and the last layer is the output layer, as can be seen Figure 12.

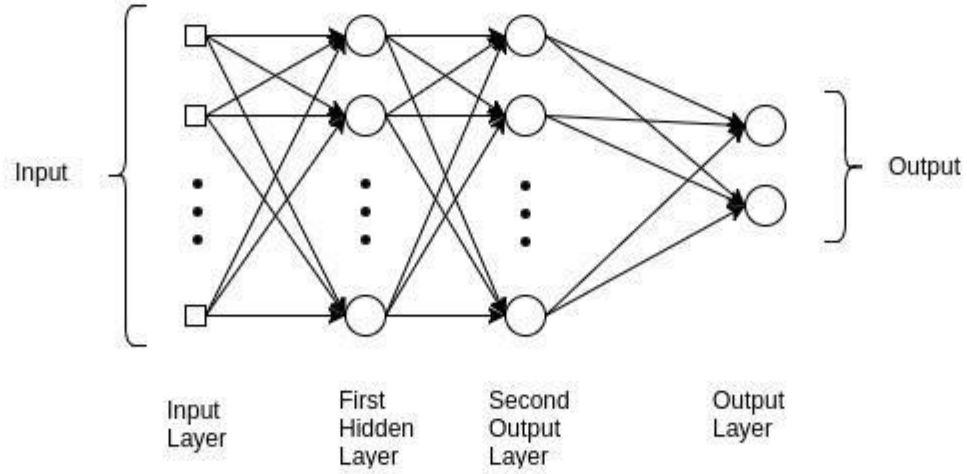


Figure 12: Multilayer perceptron-NN

MLP networks are composed of many functions that are chained together. Each layer is represented as y = f(WxT + b) where f represents the activation function, W represents the set of parameters, or weights, in the layer, and x represents the input vector. To use this algorithm in, Python, the tensorflow and keras library needs to be imported. Keras works as an interface for TensorFlow which focuses on the inference of deep neural networks.

Training and testing the model: According to Wiley et al, (*Data Mining for Business Analytics: Concepts, Techniques, and Applications in R | Wiley*, n.d.), once the model is selected based on the problem definition and domain knowledge, it is time to feed the training set to the algorithm so that it can learn suitable parameters and features used in the data set. Validation data set is mainly used in modifying or discarding variables and includes a process of tuning model-specific hyperparameters until a satisfactory accuracy level is accomplished.

In the testing stage, a test dataset is used to verify that model using accurate features. In this part of the workflow, one should return to training the model based on the feedback to improve accuracy and desired output.

Model Evaluation and Feedback: It can be done using accuracy, precision, recall and F1-score. Generally, finding accuracy can determine whether a model is being trained correctly and how it may function. The problem with using accuracy is that it does not do well when one has a severe class imbalance, that is why precision, recall must also be considered. **Error! Reference source**

not found.Error! Reference source not found. is the formula for how accuracy is mathematically interpreted.

$$Accuracy = \frac{truepositives + truenegatives}{totalexamples}$$

Precision is helpful when false positives are high, precision can help predict positives. Let us assume a model has very low precision, which can lead to the assumption that there are a lot more delayed flights, and this may be a false conclusion. **Error! Reference source not found.** is the mathematical equation for precision.

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

The opposite of precision is recall, recall helps when the false negatives are high, as false negatives can also be misleading. **Error! Reference source not found.** is the mathematical equation for recall:

$$Recall = \frac{truepositives}{truepositives + falsenegatives}$$

When modelling, machine learning algorithms assume that the data is evenly distributed within classes. If the imbalanced data is not taken care of, the model may predict high accuracy, but this will not have any value to the objective. According to (James et al., 2000) the F1-score is simply the harmonic mean of precision (PRE) and recall (REC). The balance between precision and recall can be found using the F1-score metric **Error! Reference source not found.**, which is beneficial toward imbalanced datasets.

$$\text{F1 Score} = \frac{\text{2 x (Precision x Recall)}}{\text{Precision + Recall}}$$

Interpretable Machine Learning:

Application of Machine learning is now widespread but ML models are still considered as a black box which is a barrier to the adoption of machine learning. Why can a ML model make certain prediction? That is one of the big questions in the implementation of ML. Interpretability of those prediction can help to increase the trust, to select better models and to reduce the bias that exist in the data set. Figure below is a pictorial explanation of the importance of interpretable machine learning. (*Interpretable Machine Learning - Christoph Molnar - Google Books*, n.d.) ((Masís, 2021)

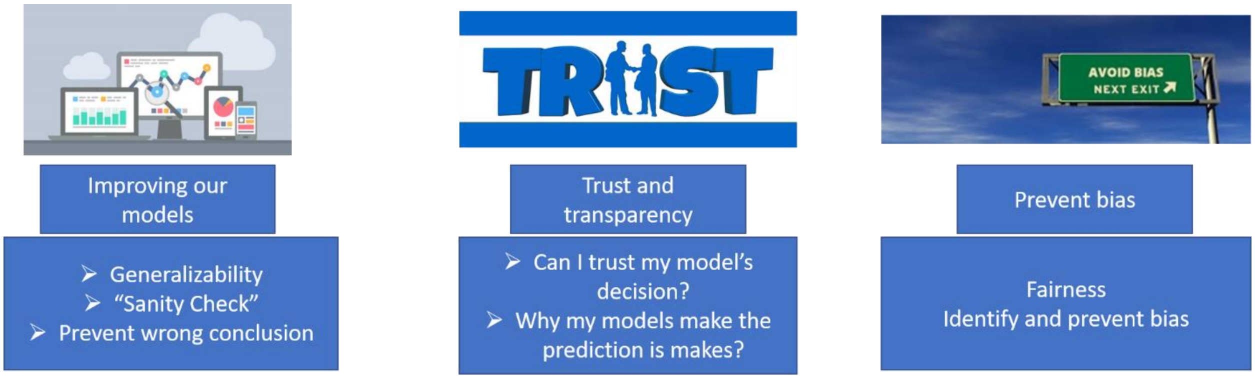


Figure 13: Importance of interpretable ML

Lime stands for Local Interpretable Model-Agnostic Explanations. LIME allows end-users to interpret predictions and take action. It is model-agnostic, implying it can be applied to any machine learning model.

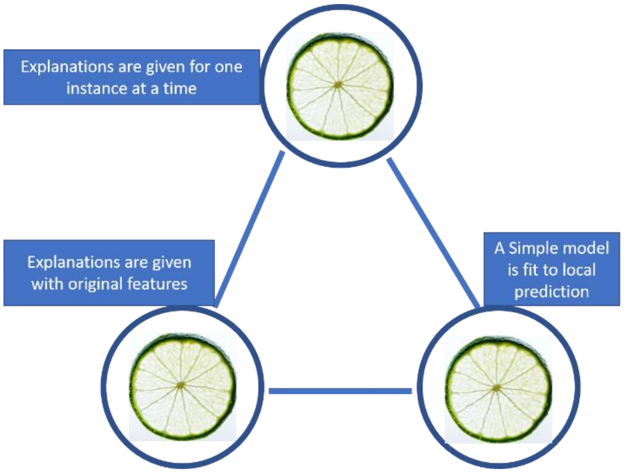

Explanations are given for one instance at a time
Explanations are given with original features
A Simple model is fit to local prediction

Figure 14: schematic of how LIME works

Above Figure is a schematic of how LIME works. The technique attempts to understand the model by studying the input of data and recognizing how the predictions change. In simple words, LIME assumes a black-box machine learning model and examines the relationship between input and output. It helps understand feature importances on a dataset level. Also, it allows verification of the problem statement, but when using LIME, it is important to accurately interpret the output.

The way LIME gives explanation of the model is by approximating the black box model for each prediction to explain, permute the observation n times, then let the complex model to predict the outcome of all permuted observations. It calculates the distance from all permutations to the original observation. Followed by converting the distance to a similarity score. Select m features best describing the complex model outcome from the permuted data. Then it fits a simple model to the permuted data, explaining the complex model outcome with the m features from the permuted data weighted by its similarity to the original observation. Finally, extract the feature weights from the simple model and use these as explanations for the complex models local behavior. Figure below explain steps that LIME uses for the interpretation (Ribeiro et al., 2016).

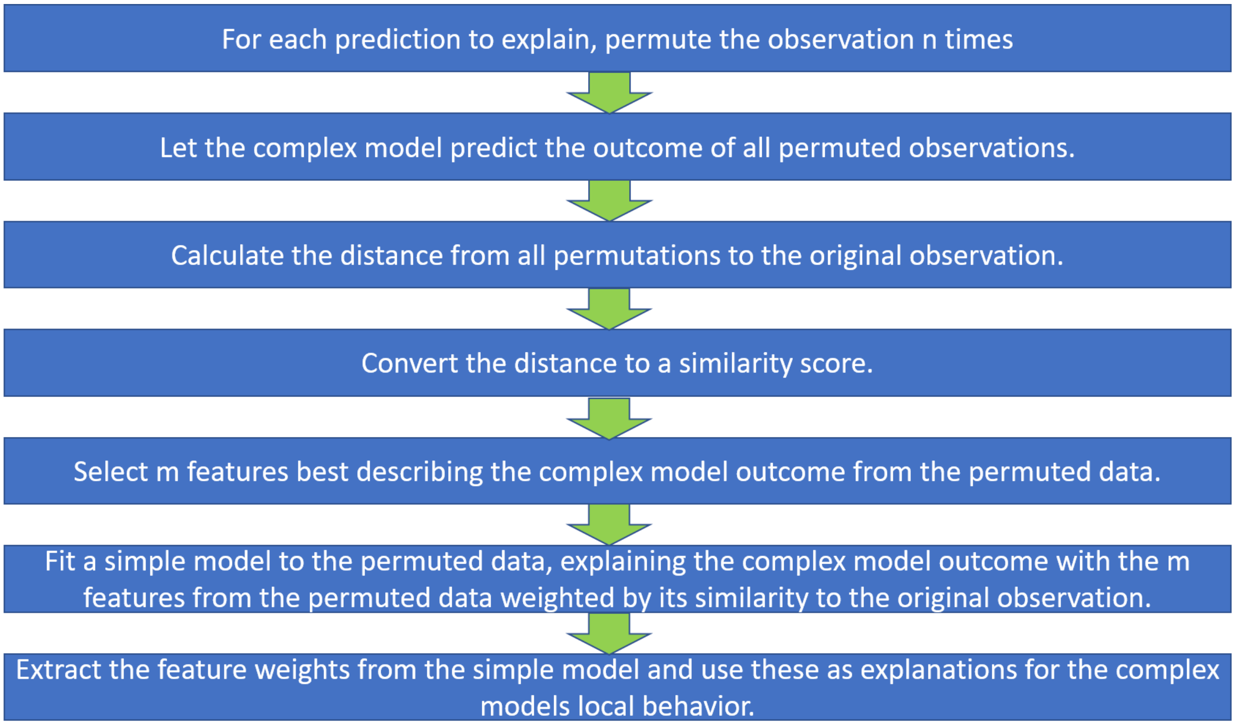

For each prediction to explain, permute the observation n times
Let the complex model predict the outcome of all permuted observations.
Calculate the distance from all permutations to the original observation.
Convert the distance to a similarity score.
Select m features best describing the complex model outcome from the permuted data.
Fit a simple model to the permuted data, explaining the complex model outcome with the m features from the permuted data weighted by its similarity to the original observation.
Extract the feature weights from the simple model and use these as explanations for the complex models local behavior.

Figure 15: Steps that LIME uses for the interpretation

Chapter 3

DATA GATHERING PREPROCESSING AND EXPLORATION

American Airlines, Inc. (AA) is an American airline carrier that is based on Fort Worth, Texas. It is arguably the world's largest airline on grounds of revenue, fleet size, and scheduled passenger kilometers flown [2]. Hence, in addition to international flights, it is observed to be quite obvious for passengers to prefer American Airlines for domestic flights too. However, flight delays concerned with American Airlines often seem sudden and unprecedented and these delays cause passengers to lose their trust in such a widely-known and internationally recognized airline. Therefore, an Intelligent and Automated Prediction System is a must in this case to predict possible airline delays. The model that will be discussed takes the flight details regarding American Airlines; data was gathered from the Bureau of Transportation Statistics (BTS) for 2016-2019 for flight that departed for DFW from following Origin Airport.

1. Los Angeles International Airport (LAX)
2. O'Hare International Airport (ORD)
3. Phoenix Sky Harbor International Airport (PHX)
4. Miami International Airport (MIA)
5. Charlotte Douglas International Airport (CLT)
6. Denver International Airport (DEN)
7. Philadelphia International Airport (PHL)
8. Ronald Reagan Washington National Airport (DCA)
9. John F. Kennedy International Airport (JFK)
10. Hartsfield-Jackson Atlanta International Airport (ATL)
11. LaGuardia Airport (LGA)

Some of the selected Airports were the operating airports (Hub) and some are busiest airports of USA.

Southwest Airlines Co. (WN) is the world's largest low-cost carrier airline based on Dallas. Data from 2016-19 was collected from BTS for the flights that are departed for Dallas Love Field Airport (DAL),one of the operating airport of Southwest, from following origin airports.

1. Texas. William P. Hobby Airport (HOU)
2. Denver International Airport (DEN)

3. Chicago Midway International Airport (MDW)
4. Phoenix Sky Harbor International Airport (PHX)
5. McCarran International Airport (LAS)
6. Hartsfield-Jackson Atlanta International Airport (ATL)
7. Los Angeles International Airport (LAX)
8. Baltimore/Washington International Thurgood Marshall Airport (BWI)
9. Orlando International Airport (MCO)
10. Oakland International Airport (OAK)

For both cases data will be analyzed and arrival delay prediction will be made, which means that it will determine whether it will arrive at the concerned airport on-time or not. So, this is a binary classification problem.

Some of the following features will be considered as that will play a key role in determining the flight delay. Each feature has a different measurable property that is used while training the model. Domain knowledge of these features should be present to understand the dataset for a better outcome.

Below are the flight data features and its description from the year 2016 and 2019, which has been extracted from the Bureau of Transportation Statistics. There were 86 features found but by conducting literature review and the need to reduce data complexity, only 35 features will be used and are listed below in below.

Table 1:Features with description

Feature Information		
Time Period		
1.	YEAR :	Year
2.	QUARTER:	Quarter (1-4)
3.	MONTH:	Month
4.	DAY_OF_MONTH:	Day of Month
5.	DAY_OF_WEEK:	Day of Week

Airline		
6.	OP_UNIQUE_CARRIER:	Unique Carrier Code. When the same code has been used by multiple carriers, a numeric suffix is used for earlier users, for example, PA, PA(1), PA(2). Use this field for analysis across a range of years.
7.	OP_CARRIER_FL_NUM:	Flight Numb
Origin		
8.	ORIGIN:	Origin Airport Analysis
Destination		
9.	DEST:	Destination Airport Analysis
Departure Performance		
10.	CRS_DEP_TIME:	CRS Departure Time (local time: hhmm)
11.	DEP_TIME:	Actual Departure Time (local time: hhmm)
12.	DEP_DELAY:	The difference in minutes between scheduled and actual departure time. Early departures show negative numbers.
13.	DEP_DELAY_NEW:	The difference in minutes between scheduled and actual departure time. Early departures are set to 0.
14.	DEP_DEL15:	Departure Delay Indicator, 15 Minutes or More (1=Yes)
15.	TAXI_OUT:	Taxi Out Time, in Minutes
16.	WHEELS_OFF:	Wheels Off Time (local time: hhmm)
	Arrival Performance	
17.	WHEELS_ON:	Wheels On-Time (local time: hhmm)
18.	TAXI_IN:	Taxi In Time, in Minutes
19.	CRS_ARR_TIME:	CRS Arrival Time (local time: hhmm)
20.	ARR_TIME:	Actual Arrival Time (local time: hhmm)
21.	ARR_DELAY:	The difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers.
22.	ARR_DELAY_NEW:	The difference in minutes between scheduled and actual arrival time. Early arrivals are set to 0.
23.	ARR_DEL15:	Arrival Delay Indicator, 15 Minutes or More (1=Yes)
Cancellations and Diversions		
24.	CANCELLED:	Cancelled Flight Indicator (1=Yes)

25.	CANCELLATION_CODE:	Specifies The Reason For Cancellation
26.	DIVERTED:	Diverted Flight Indicator (1=Yes)
Flight Summaries		
27.	CRS_ELAPSED_TIME:	Elapsed Time of Flight, in Minutes
28.	ACTUAL_ELAPSED_TIME:	Elapsed Time of Flight, in Minutes
29.	AIR_TIME:	Flight Time, in Minutes
	Flights Number of Flights Analysis	
30.	DISTANCE:	Distance between airports (miles)
Cause of Delay		
31.	CARRIER_DELAY:	Carrier Delay, in Minutes
32.	WEATHER_DELAY:	Weather Delay, in Minutes
33.	NAS_DELAY:	National Air System Delay, in Minutes
34.	SECURITY_DELAY:	Security Delay, in Minutes
35.	LATE_AIRCRAFT_DELAY:	Late Aircraft Delay, in Minutes

For this thesis, 2016-2019 data was used, and the listed features with descriptions are given in Table 1.

Data Cleaning: Since this is a binary classification problem, let us represent 0 for ARR_TIME arrival on and 1 for delayed arrival for the above features. Since it is a large dataset (153975 entries with a total of 35 features), additional steps need to be taken by studying each feature and missing values must be considered. For instance, when executing df1.isna().sum(), where df1 is our dataset the missing count of values are listed below.

```
 0   YEAR                 153975 non-null  int64
 1   QUARTER              153975 non-null  int64
 2   MONTH                153975 non-null  int64
 3   DAY_OF_MONTH         153975 non-null  int64
 4   DAY_OF_WEEK          153975 non-null  int64
 5   OP_UNIQUE_CARRIER    153975 non-null  object
 6   OP_CARRIER_FL_NUM    153975 non-null  int64
 7   ORIGIN               153975 non-null  object
 8   DEST                 153975 non-null  object
 9   CRS_DEP_TIME         153975 non-null  int64
 10  DEP_TIME             151722 non-null  float64
 11  DEP_DELAY            151718 non-null  float64
 12  DEP_DELAY_NEW        151718 non-null  float64
 13  DEP_DEL15            151718 non-null  float64
 14  TAXI_OUT             151612 non-null  float64
 15  WHEELS_OFF           151612 non-null  float64
 16  WHEELS_ON            151578 non-null  float64
 17  TAXI_IN              151578 non-null  float64
 18  CRS_ARR_TIME         153975 non-null  int64
 19  ARR_TIME             151578 non-null  float64
 20  ARR_DELAY            151026 non-null  float64
 21  ARR_DELAY_NEW        151026 non-null  float64
 22  ARR_DEL15            151026 non-null  float64
 23  CANCELLED            153975 non-null  float64
 24  CANCELLATION_CODE    2367 non-null    object
 25  DIVERTED             153975 non-null  float64
 26  CRS_ELAPSED_TIME     153975 non-null  float64
 27  ACTUAL_ELAPSED_TIME  151026 non-null  float64
 28  AIR_TIME             151026 non-null  float64
 29  DISTANCE             153975 non-null  float64
 30  CARRIER_DELAY        31058 non-null   float64
 31  WEATHER_DELAY        31058 non-null   float64
 32  NAS_DELAY            31058 non-null   float64
 33  SECURITY_DELAY       31058 non-null   float64
 34  LATE_AIRCRAFT_DELAY  31058 non-null   float64
```

CANCELLED and DIVERTED feature are dropped as those flight doesn’t arrive at the airports.

BTS listed following five features for reasons for delay:

- CARRIER_DELAY
- WEATHER_DELAY
- NAS_DELAY
- SECURITY_DELAY
- LATE_AIRCRAFT_DELAY

Before making any decisions related to these features, how much information is available must be determined. By executing the Count for “NaN” or missing values in DataFrame, it was found that 122917 values were missing among all of the five features and the following result was obtained: the percentage of valid data was found to 20.1708% and the percentage of missing values was found to be 79.829%. Therefore, all “NaN” values were replaced to 0 for the five delay reasons. DEP_DEL15 and ARR_DEL15 are dropped as we are interested in just whether the flight will be delayed or not. As DEP_TIME (departure time) and ARR_TIME (arrival time) features seemed vague because it does not comprise dates, those features will be converted to the quarter of the day.

In the following features TAXI_IN, TAXI_OUT, DEP_DELAY, DEP_DELAY_NEW, ARR_DELAY, ARR_DELAY_NEW the Count for “NaN” or missing values were executed in DataFrame and it was found that 2397, 2363, 2257, 2257, 2949, 2949 were missing, respectively. Since these missing values cannot be retrieved the “NaN” will be replaced by the mean or median.

For the feature CANCELLATION_CODE, by executing the Count for “NaN” or missing values in DataFrame, it was found that 151608 values were missing, this feature consists of three categories A, B, and C, where A represents Carrier cancellation, B represents Weather delay, and C represents National Air System delay. Since this is a categorical feature, it cannot be replaced by using mean or median, instead, it was replaced by the most appeared category B. CRS_DEP_TIME, and CRS_ARR_TIME is usually a categorical value, however the current format results in too many columns, therefore, time is split into quadrants.

The feature ARR_DELAY, arrival delay reveals the difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers, since this will not be feasible when modeling, this data will not be used. Therefore, by comparing the value of each row for the "ARR_DELAY" feature, if the total delay is zero or less, it will assign "0", this is denoted as minus-- arrive early in the dataset, otherwise it will be assigned as "1". Once that was completed, that feature was then added to a new feature (column), denoted as "ARR_FLIGHT_STATUS".

Similarly, for Departure Delay, when comparing the value of each row for the "DEP_DELAY" feature (or column), if the total delay zero or less (minus-- arrive early) it will assign "0", otherwise, it will be assigned as "1". Once that was completed, that feature was then added to a new feature (column) " DEP _FLIGHT_STATUS".

The same will be done for the Cause of delay features such as CARRIER_DELAY, by comparing the value of each row for the "CARRIER_DELAY" feature (or column), if the total delay zero or less (minus-- arrive early) it will be assigned as "0" otherwise, it will be assigned as "1" and feature (column) called "CARRIER_DELAY_STATUS" will be added and the same process will be executed on WEATHER_DELAY, NAS_DELAY, SECURITY_DELAY, and LATE_AIRCRAFT_DELAY

After having added these additional features, there are a total of 40 features as listed below.

```
Data columns (total 40 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   YEAR                   151026 non-null  int64  
 1   QUARTER                151026 non-null  int64  
 2   MONTH                  151026 non-null  int64  
 3   DAY_OF_MONTH           151026 non-null  int64  
 4   DAY_OF_WEEK            151026 non-null  int64  
 5   OP_UNIQUE_CARRIER      151026 non-null  object 
 6   OP_CARRIER_FL_NUM      151026 non-null  int64  
 7   ORIGIN                 151026 non-null  object 
 8   DEST                   151026 non-null  object 
 9   CRS_DEP_TIME           151026 non-null  int64  
 10  DEP_DELAY              151026 non-null  float64
 11  DEP_DELAY_NEW          151026 non-null  float64
 12  DEP_DEL15              151026 non-null  float64
 13  TAXI_OUT               151026 non-null  float64
 14  WHEELS_OFF             151026 non-null  int64  
 15  WHEELS_ON              151026 non-null  int64  
 16  TAXI_IN                151026 non-null  float64
 17  CRS_ARR_TIME           151026 non-null  int64  
 18  ARR_DELAY              151026 non-null  float64
 19  ARR_DELAY_NEW          151026 non-null  float64
 20  ARR_DEL15              151026 non-null  float64
 21  CANCELLED              151026 non-null  float64
 22  CANCELLATION_CODE      151026 non-null  object 
 23  DIVERTED               151026 non-null  float64
 24  CRS_ELAPSED_TIME       151026 non-null  float64
 25  ACTUAL_ELAPSED_TIME    151026 non-null  float64
 26  AIR_TIME               151026 non-null  float64
 27  DISTANCE               151026 non-null  float64
 28  CARRIER_DELAY          151026 non-null  float64
 29  WEATHER_DELAY          151026 non-null  float64
 30  NAS_DELAY              151026 non-null  float64
 31  SECURITY_DELAY         151026 non-null  float64
 32  LATE_AIRCRAFT_DELAY    151026 non-null  float64
 33  ARR_FLIGHT_STATUS      151026 non-null  int64  
 34  DEP_FLIGHT_STATUS      151026 non-null  int64  
 35  CARRIER_DELAY_STATUS   151026 non-null  int64  
 36  WEATHER_DELAY_STATUS   151026 non-null  int64  
 37  NAS_STATUS             151026 non-null  int64  
 38  SECURITY_STATUS        151026 non-null  int64  
 39  LATE_AIRCRAFT_STATUS   151026 non-null  int64  
```

The following features ARR_DELAY, DEP_DELAY, CARRIER_DELAY, WEATHER_DELAY, NAS_DELAY, SECURITY_DELAY, LATE_AIRCRAFT_STATUS, 'DEP_DELAY_NEW', 'DEP_DEL15','ARR_DELAY_NEW', 'ARR_DEL15','CANCELLED','CANCELLATION_CODE', and 'DIVERTED' are duplicated and therefore, require to be dropped, which then results in a total of 23 features.

Certain subsets of the 26 remaining features were found to have a high correlation among them. Correlation is a technique that can indicate whether and how strongly pairs of variables are related. Correlations are useful because they can help in determining the relationships amongst variables, and can also be used to make predictions. A correlation coefficient places a value on the relationship, this index has a value between -1 and 1, where 0 means there is no relationship between the variables, and -1 or 1 suggests that there is a high negative or positive correlation, respectively (James et al., 2000). It can be illustrated using a heat map; a two-dimensional graphical interpretation of data where the values are represented in colors in a matrix. below is a heat map of the 23 features.

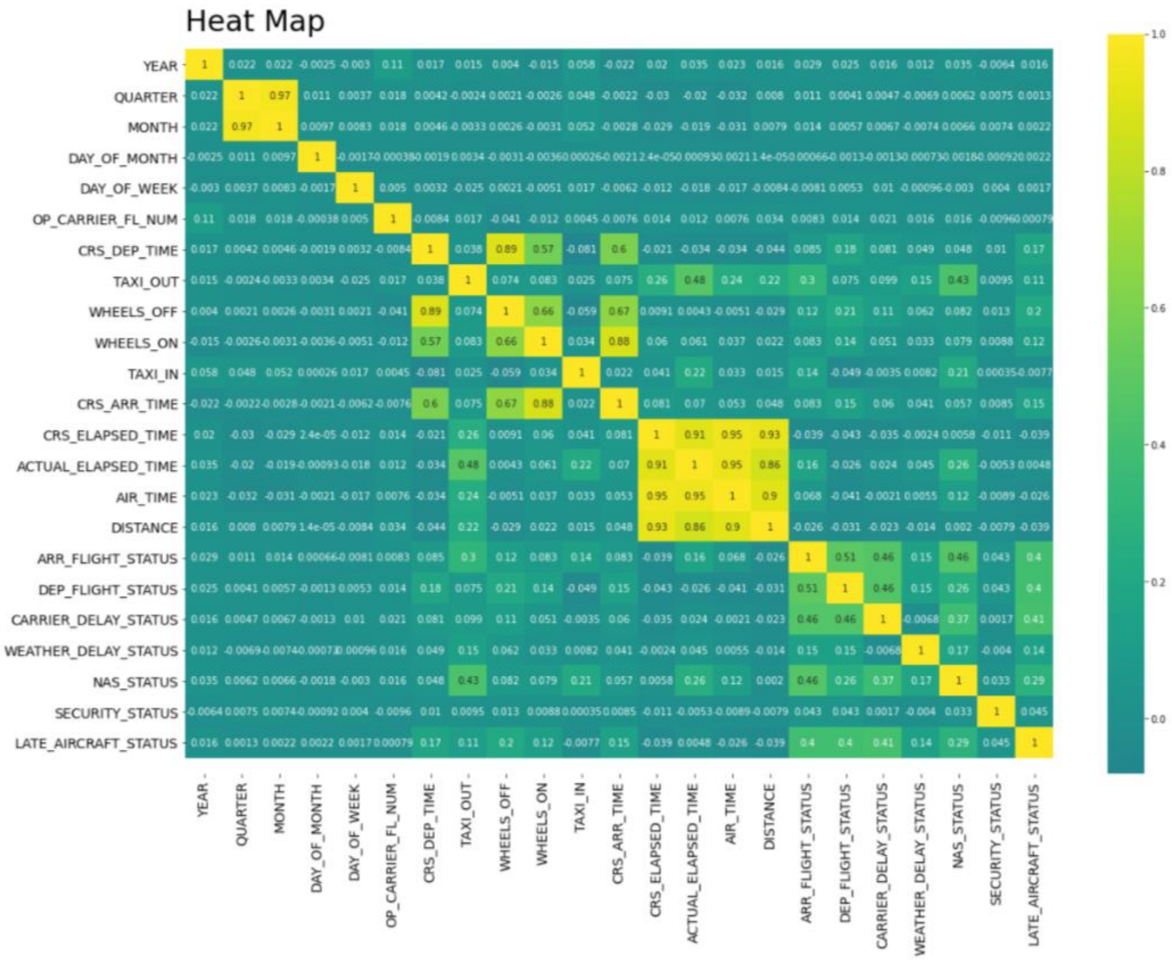


Figure 16:Heat Map

From the color on the indicator, it can instantly be seen that certain features have a higher correlation than others, for example, MONTH vs. QUARTER has a high correlation of 0.97, one of these features can be dropped but it is kept to help visualize the correlation in data exploratory.

EDA: Exploratory Data Analysis for American Airlines

In Exploratory Data Analysis (EDA), the large number of rows and columns which was formatted during the preprocessing steps will help in visualizing, summarizing, and interpreting the data. EDA is one of the steps in data science that provides certain insights and meaningful patterns which is essential not only for data scientists but also for business aspects. EDA reveals information about the content without having to make any assumptions, that is why data scientists use this process to understand what kind of model can be created for further analysis, according to (Suresh Kumar Mukhiya and Usman Ahmed, n.d.). In Python, the libraries which will be used with visualizing and informative statistical graphics include matplotlib and Plotly, and Seaborn which is a Python data visualization library based on matplotlib.

Using the dataset, which was limited to US domestic flight for AA and the following airports: LAX, ORD, PHX, MIA, CLT, DEN, PHL, DCA, JFK, ATL, LGA, the objective is to find the total number of AA flights which arrived at DFW during the year 2016-2019. It is important to note that the 2020 dataset was not used due to COVID-19, as a different trend may appear in reasons for delay, which can be analyzed during future work. The bar chart below shows the exact number of flights which arrived at DFW, here the x-axis represents the name of the origin airport and the y-axis represents the number of flights.

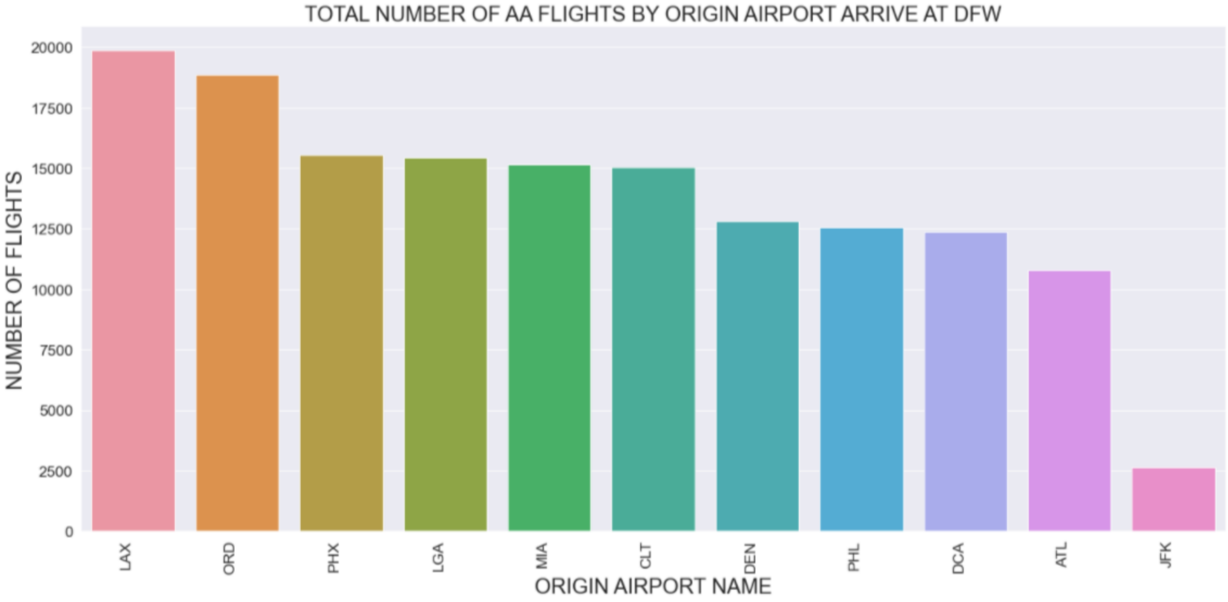


Figure 17: Total number of AA flights that arrived at DFW

The above graph illustrates the number flights that arrived at DFW from the following airports LAX, ORD,PHX , LGA, MIA, CLT, DEN, PHL, DCA, ATL, and JKF, respectively. The purpose is to find how many of these flights were delayed, and then the reason for the delay can be determined. By keeping the x-axis and y-axis the same as shown above and grouping the origin and flight status in descending order, shows the flights arriving from ORD to DFW have the highest number of delays. According to this graph, a trend that can be seen is that the higher the number of flights arriving from a particular airport, the higher the possibility of the flights being delayed. However, that may be true for many of the airports, the graph below indicates that ORD has more delayed flights than LAX, similarly, that can be observed in other cases.

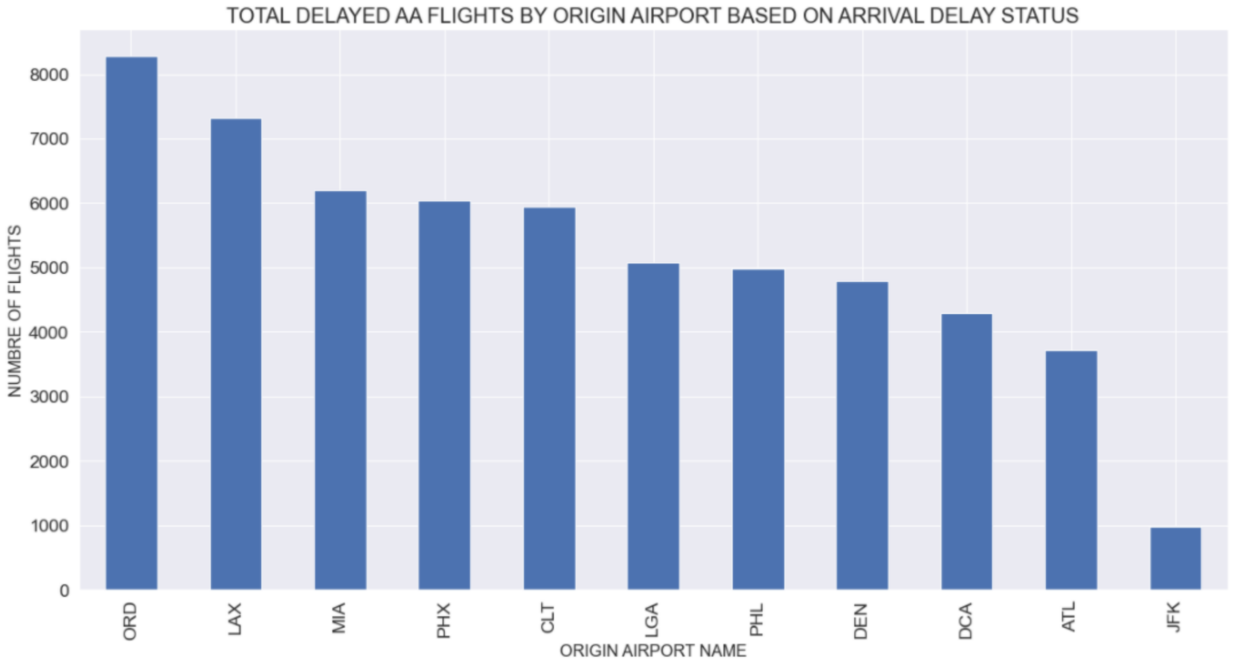


Figure 18: Total Delayed AA Flights by origin airport based in arrival delay status

To get a greater understanding, a new engineered binary column ARR_FLIGHT_STATUS was added using the column ARR_DELAY where it represents flights being delayed or not, where “0” indicates that flights arrived on time, and “1” indicates that flights arrived late. From the new engineered feature, the overall percentage of delayed flights per Origin Airport showed approximately 61.8% of flights from the dataset have arrived on time and 38.2 % were delayed. Departure Delay involves the plane taking off later than its departure time, making the chances higher for the flight to be arriving late at its destination. A new engineered binary column DEP_DELAY_STATUS was added using the column DEP_DELAY where it represents flight being delayed or not, “0” indicating flight departure to be on time and “1” for flights being departed late.

below is a pie-chart showing the Departure of Flight Status, where it is illustrated that 61.2 % of flights departed on time and 38.4% shows flights that departed late. The two pie charts show some correlations which reveal if the flight was delayed during departure, there might be a possibility it might arrive late. From both chart, one can conclude that the data in not highly imbalanced.

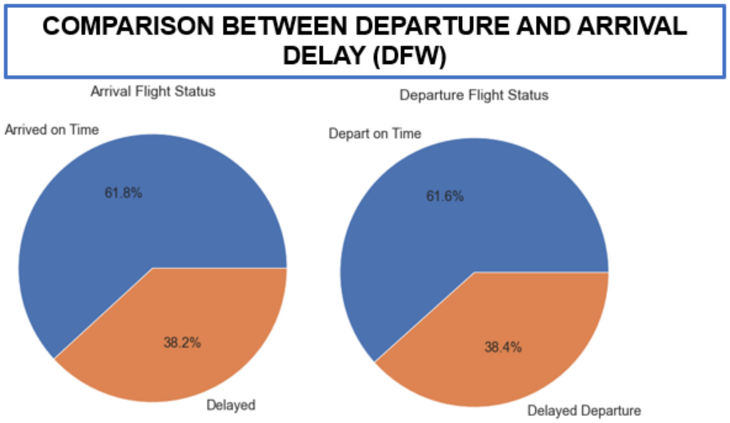


Figure 19:Comparison pie char between arrival and delay status

The below Bar-chart: Total number delayed of AA flights by Origin shows ORD has the highest number of delays. It is interesting to note that although in this graph it can be observed that JFK has the least number of delayed flights, when looking into the bar-chart below, which depicts the percentage of delayed arrival flights by airport, the results shows that JFK has a greater percentage of delayed arrival flights. For instance, when taking a flight from New York, a wiser choice between LGA and JFK would be to choose to take a flight from LGA, because the percentage of delayed flights is lower in comparison to the percentage of delayed flights at JFK.

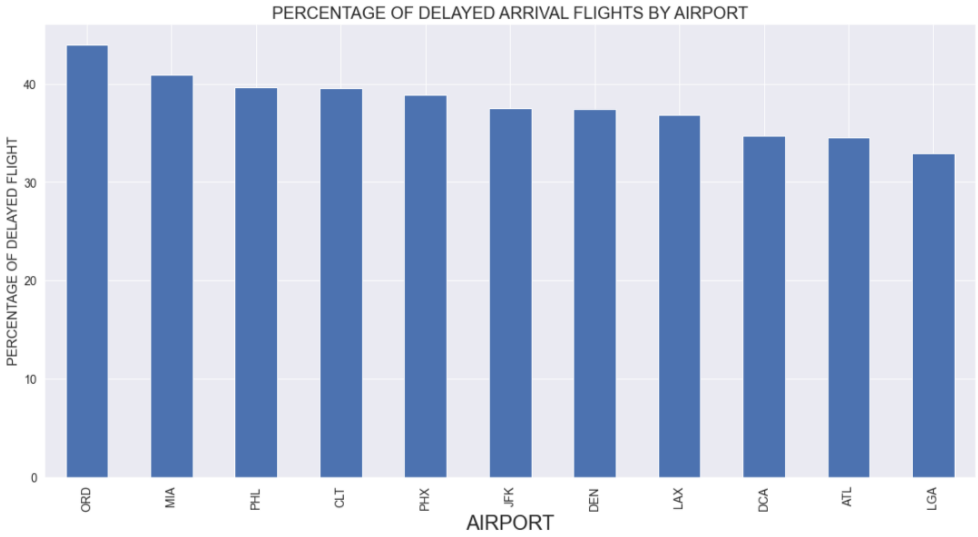


Figure 20:Percentage of delayed arrival flights of airport

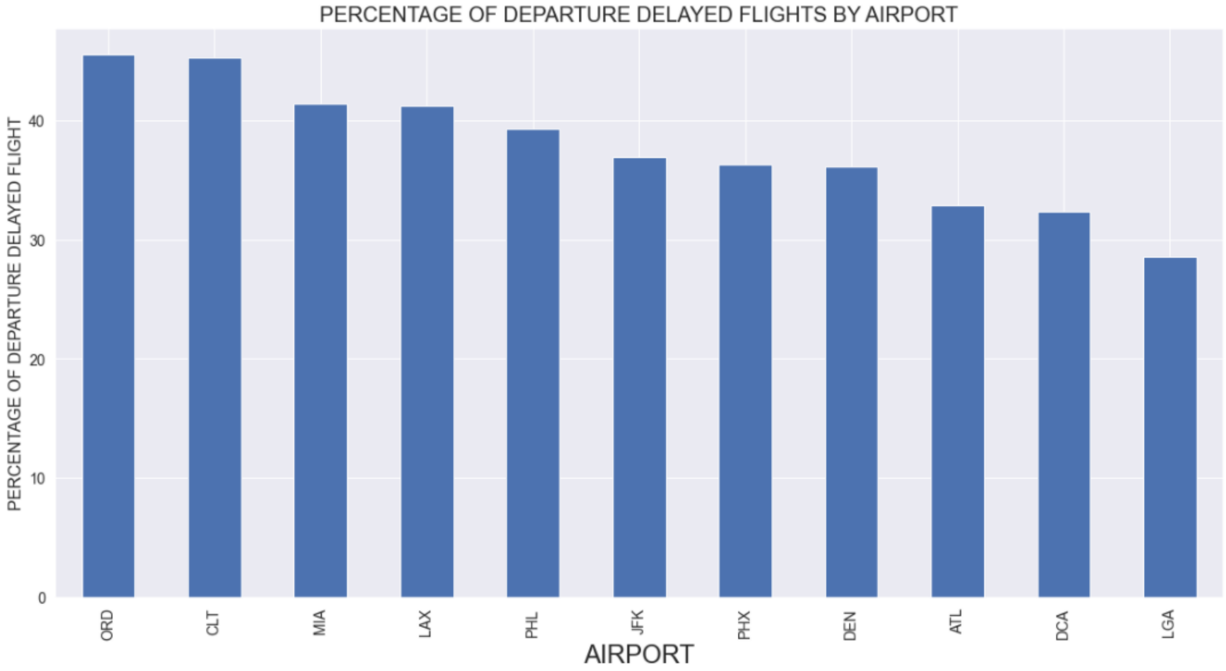


Figure 21: Percentage of departure delayed flights by airport

By looking into the percentage Figure 20,Figure 21, the below bar chart where the total delayed minutes by the airport is shown, the x-axis represents the airport and the y-axis represents the total delayed minutes, interesting results were revealed, it can be observed that the total number of minutes delayed is very low for LGA though the highest number of flights depart from LGA to DFW.

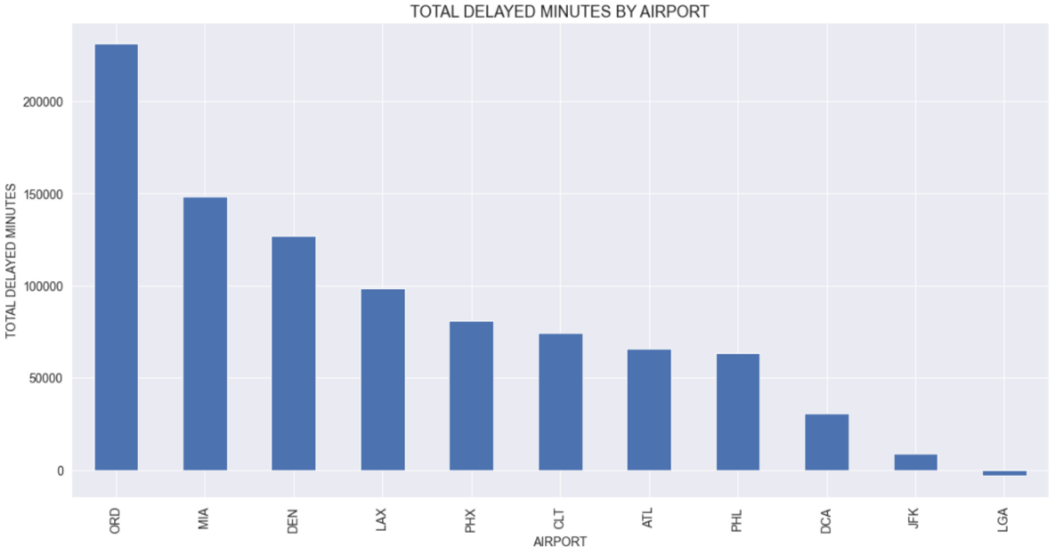


Figure 22:Total delays in minutes by airport

According to FAA, (*Airlines For America | Annual U.S. Impact of Flight Delays (NEXTOR Report)*, n.d.) in 2018, alone, airlines and passengers lost $28 billion due to delayed flights, presuming $47 per hour average value of a passenger's time. Looking into AA, and the 5 busiest airports, the Average Delay Time per Airport is shown below. The x-axis represents the minutes, and the y-axis represents origins (of airport).

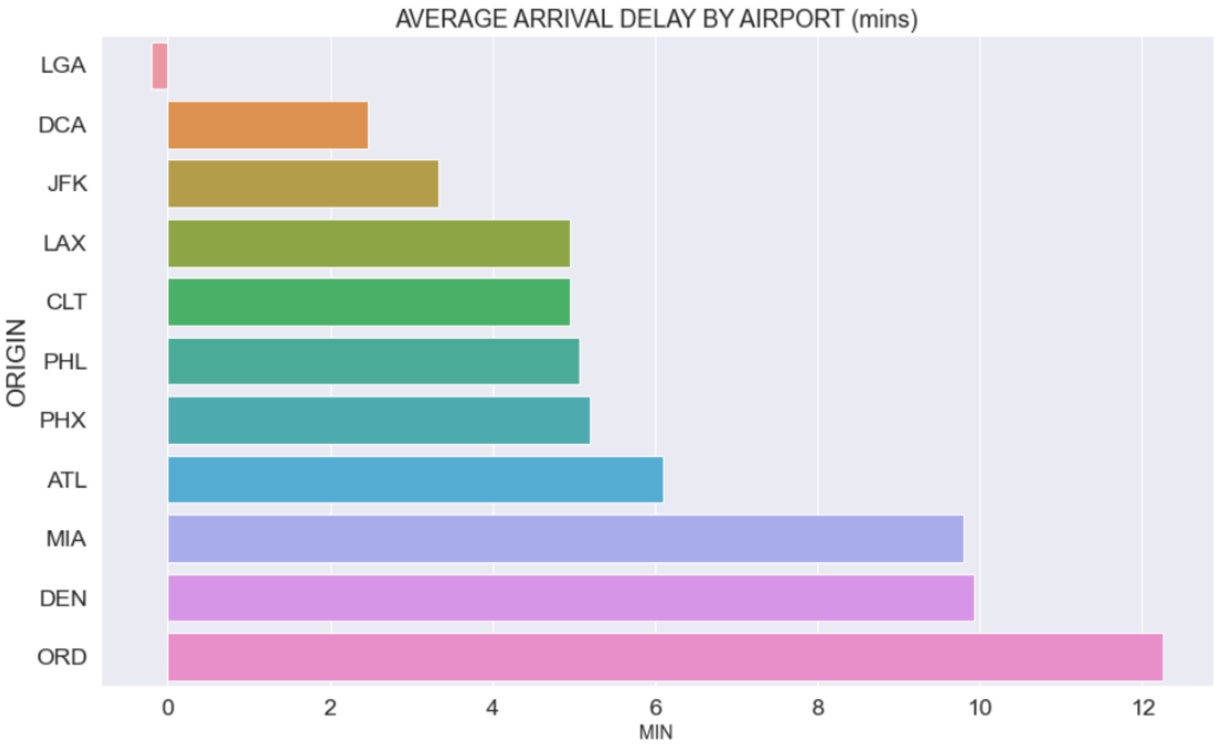


Figure 23:Average arrival delay by airport

Knowledgeable travelers will take advantage of vacation deals at popular destinations around the world. below is the bar graph showing the worst and the best months to travel to DFW based on flight delays and the origin airport, where x -axis represents the month (which is numbered from 1-12) and the y-axis represents the number of flights. The month of June is shown as the busiest month of the year to travel to DFW, however, April and September are the best months to travel to DFW to avoid delays, this is a great insight to those who would prefer to travel in conditions with less congestion and limited traffic.

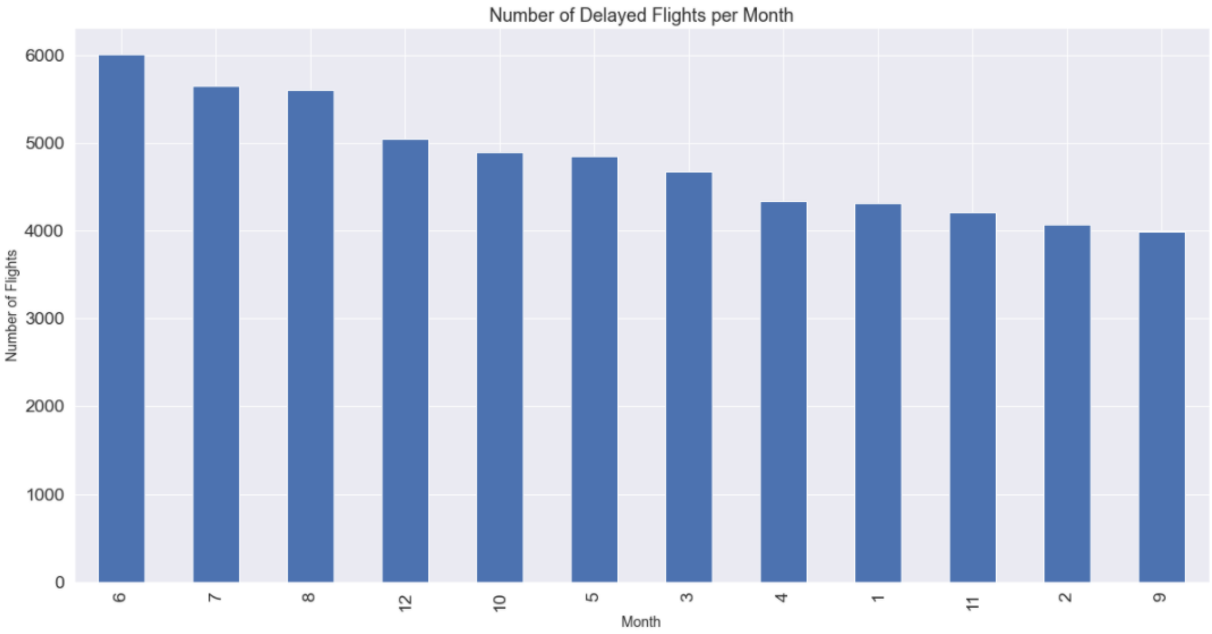


Figure 24:Number of delays flight per month

When observing the bar chart: Figure 25: Number of Delayed flights per day of week, where the x-axis represents the days of the week and the y-axis represents the number of flights, it can be noted that Thursday and Friday are the busiest when traveling to DFW. However, Tuesday and Saturday are less busy. This is due to the reason that most people have to take more days off work to fly midweek, which people usually tend to avoid. Furthermore, passengers who commute every week for work will also travel on Fridays in order to get home for the weekend and will travel on Sundays to get to work on Mondays which results on Sundays being very busy as well.

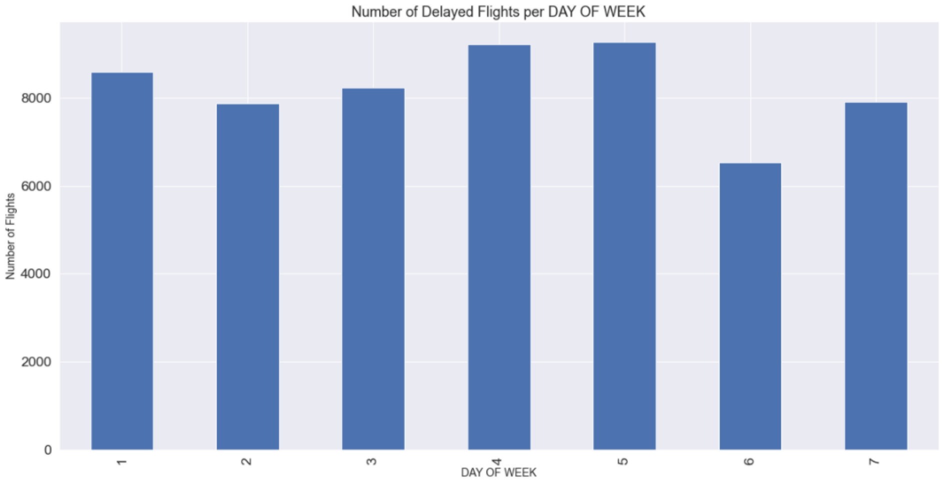


Figure 25: Number of Delayed flights per day of week

When you search for flights online, it typically offers a prediction for whether you should buy the tickets right away or wait, and the recent fare for that travel plan is also provided. The site will also predict whether the flight will be available in the next few days.

The below bar chart shows the Number of Delayed Flights Per day of the Month, the x-axis represents the Day of the Month, and the y-axis represents the number of flights. The interesting fact is the 31st has the least number of flight delays which can be due to the reason that 5 months of the year only have 30 days.

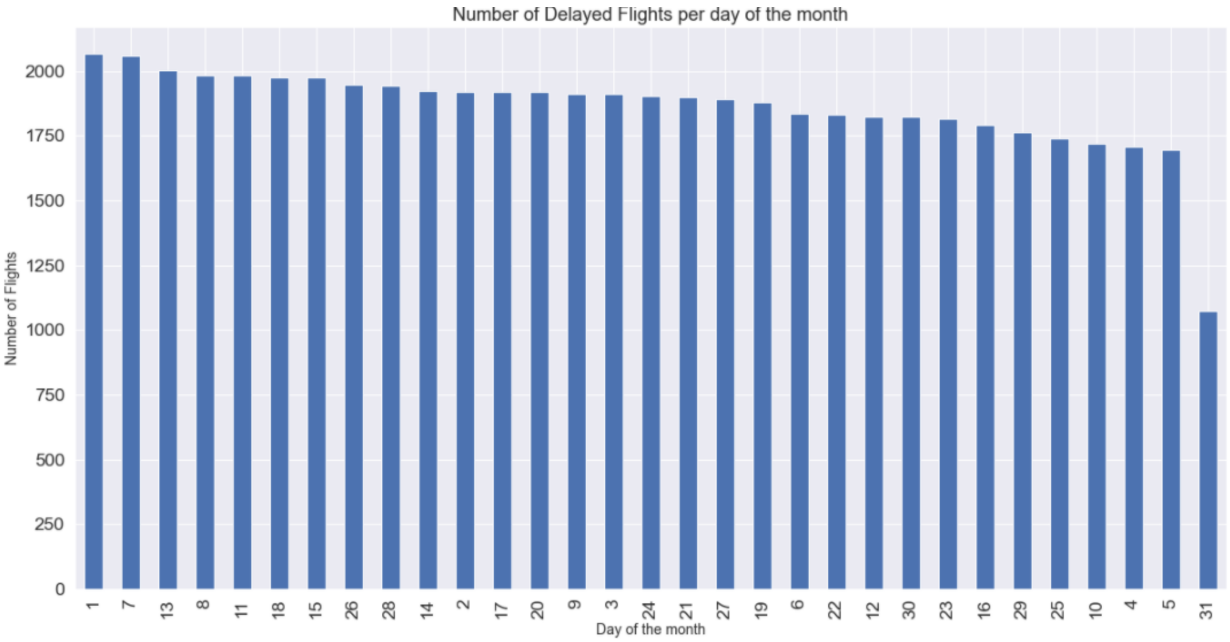


Figure 26: Number of delayed flights per day of the month

This next visualization is of the departure and the arrival delay. It shows whether departure or arrival delay has a bigger impact on the plane being delayed. Assuming that the departure of the flight was on time and the flight's arrival is late, then this means that there's another factor which may have increased the elapsed time. This plot validates the principal idea that some airlines will try to compensate for the delayed departure by reducing air time. The below graph is shown by overlaying the ARR_DELAY over the DEP_DELAY. The departure delay (DEP_DELAY) is colored in light blue, whereas the arrival delay (ARR_DELAY) is the dashed line. The results of the plot suggest that the ARR_DELAYS are generally lower than the DEP_DELAYS, and as mentioned above, this suggests that the airlines try to adjust their flight speed to try and compensate for the late departure and reduce the ARR_DELAY. Another interesting piece of information is related to LGA. Once again, the arrival delay is low. This means that even when the departure is late, the flight that leaves from LGA to DFW compensates the delay by reducing the air time of the flights to arrive on time or on average, earlier.

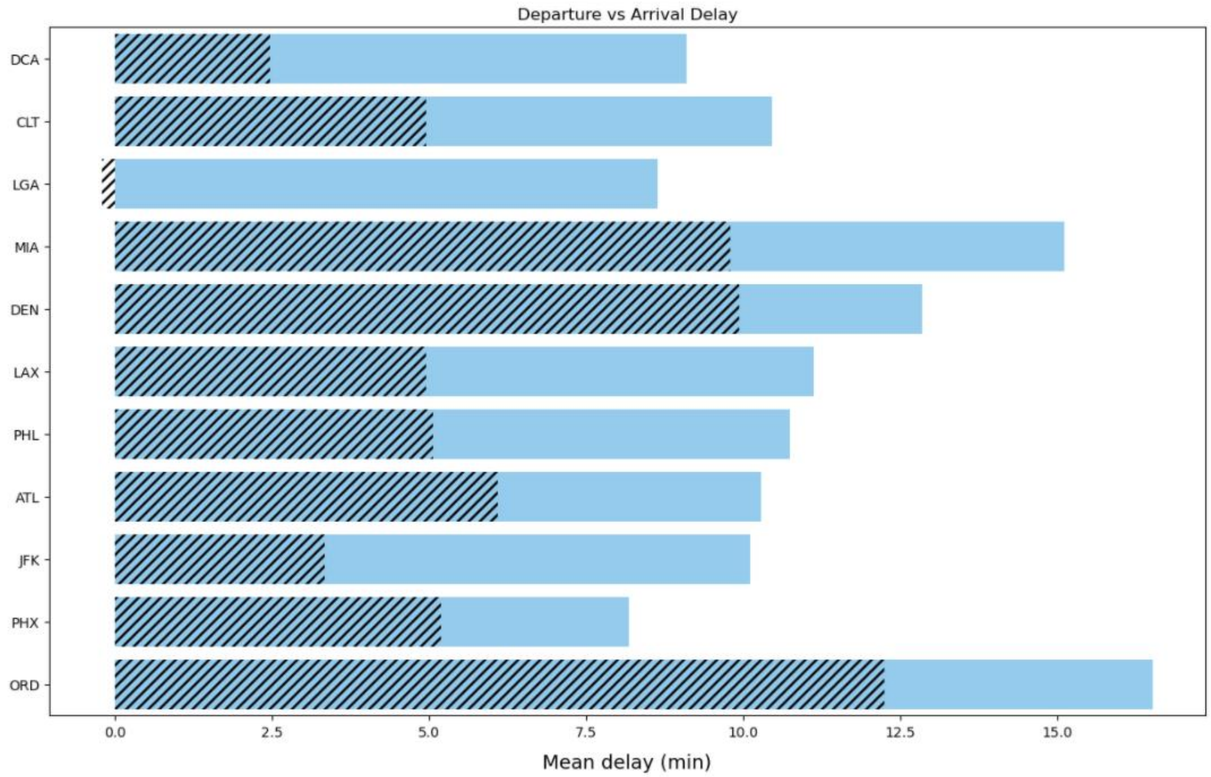


Figure 27:Departure vs arrival delay

There are five features that are causes of delay; carrier delay which involves maintenance, crew problems, aircraft cleaning, baggage loading, fueling. Weather delay which involves major meteorological conditions such as tornadoes, blizzards or hurricanes. NAS Delay which involves non-extreme weather conditions, airport operations, heavy traffic volume, air traffic control, as well as any delays that occur after the gate are usually assigned to the NAS Delay. Security delay which is caused by evacuation of a terminal as well as when reboarding of the aircraft is required. Finally, Late Aircraft delay which involves a previous flight with the same aircraft that arrived late, causing the present flight to depart late.

Below bar charts will show the causes of delay with the help of data visualizations, the bar chart below is for Total Delayed AA Flights by Origin Airport based on Carrier Delay status, where the x-axis represents the origin airport name and the y-axis represents the number of flights. The data used is normalized using the normalize function in Python, because differences in the scales across the models may cause inaccurate results when modeling. The chart suggests that most delayed flights coming from ORD to DFW are due to carrier delay whereas JFK had the least number of flights affected by carrier delays.

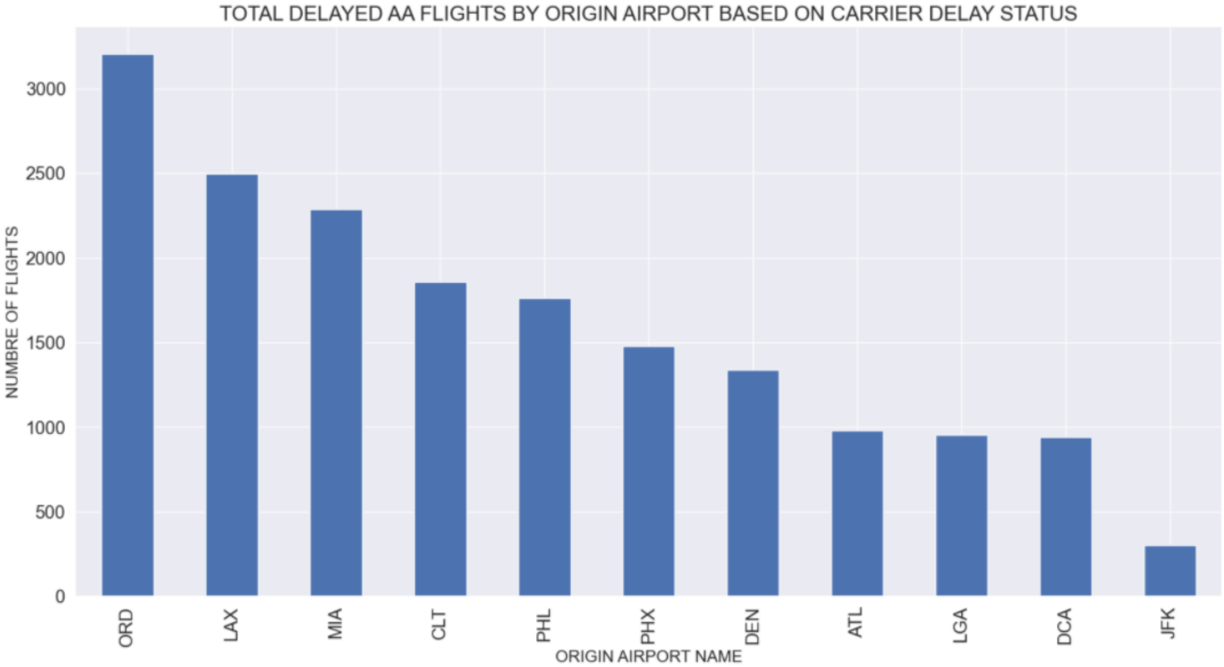


Figure 28:Total delayed AA flights by origin airport based on carrier delay

Similarly, the bar chart below represents Total Delayed AA Flights by Origin Airport based on Weather Delay status, where the x-axis represents the origin airport name and the y-axis represents the number of flights. Likewise, most delayed flights coming from ORD to DFW are due to weather delays where LAX had the least number of flights which were delayed due to weather delays.

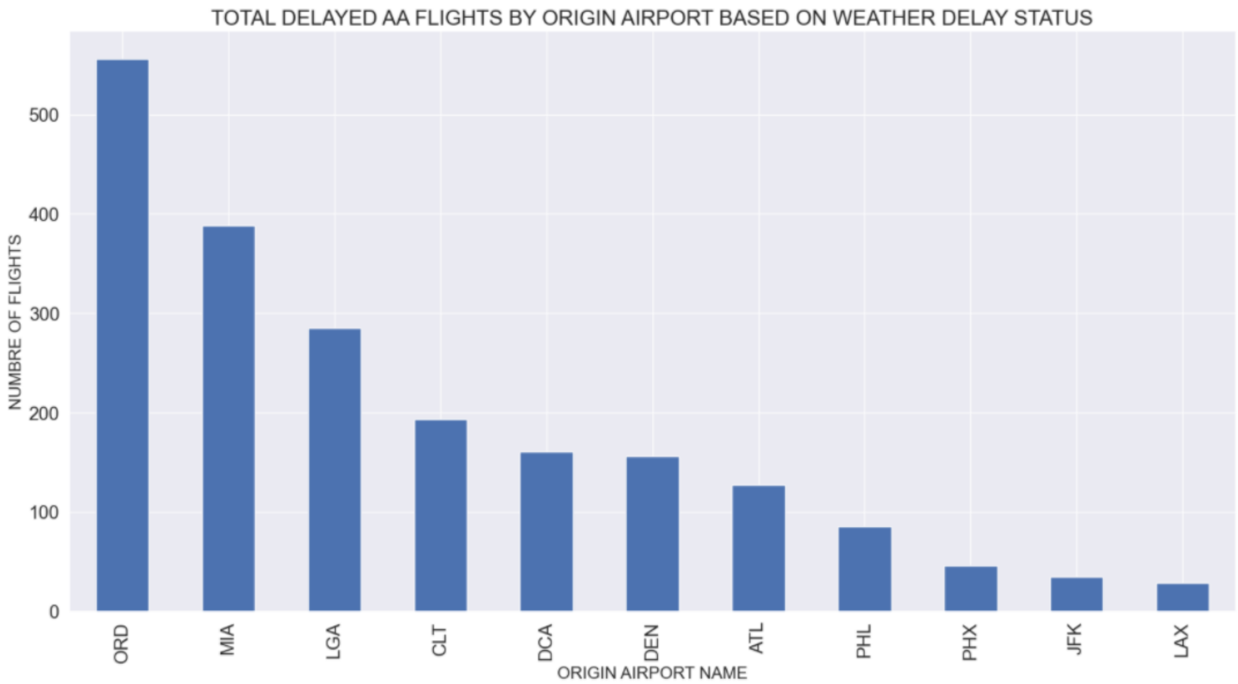


Figure 29:Total delayed AA flights by origin airport based on weather delay

NAS Delays usually include any delays that occur after Actual Gate Out (delays caused after the gate), according to the Bureau of Transportation. The bar chart below is for Total Delayed AA Flights by Origin Airport based on NAS Delay status, where the x-axis represents the origin airport name and the y-axis represents the number of flights. Most delayed flights due to NAS delay are from ORD to DFW where JFK has the least number of flights which are delayed by NAS delays.

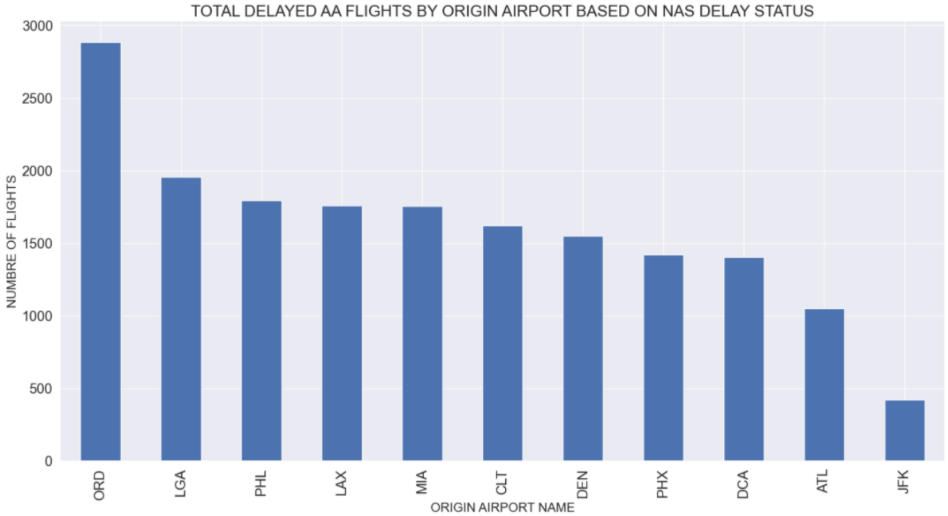


Figure 30:Total delayed AA flights by origin airport based on NAS delay

Security delays can be caused by equipment at the screening not functioning or any other occurrences of security breaches. The bar chart below is for Total Delayed AA Flights by Origin Airport based on Security Delay status, where the x-axis represents the origin airport name and the y-axis represents the number of flights. Most delayed flights due to Security delay are from CLT to DFW whereas LGA had the least number of flights delayed due to security delays. An observation that can be made is that despite ORD being a larger airport, it has a relatively lower security delay, a reason as to why that is, could be that better security measures are in place at ORD.

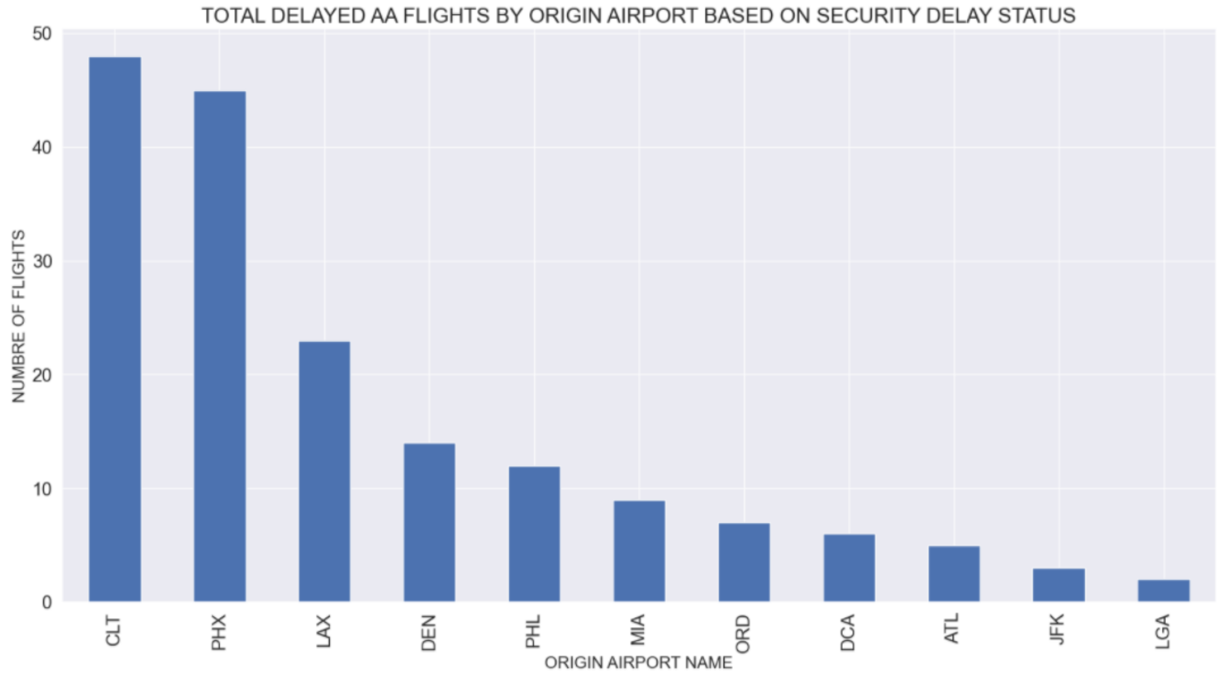


Figure 31:Total delayed AA flights by origin airport based on security delay

According to the Bureau of Transportation of Statistics, the late aircraft delay occurs when the current flight takes off later than scheduled and creates a domino effect on the following flights. The bar chart below is for Total Delayed AA Flights by Origin Airport based on Late aircraft Delay status, where the x-axis represents the origin airport name and the y-axis represents the number of flights. Most delayed flights due to Late aircraft delay are from ORD to DFW whereas JFA has the least delayed flights due to late aircraft delays. This may be caused by the reason that flights coming to ORD are being delayed, leading to flights leaving ORD also being delayed.

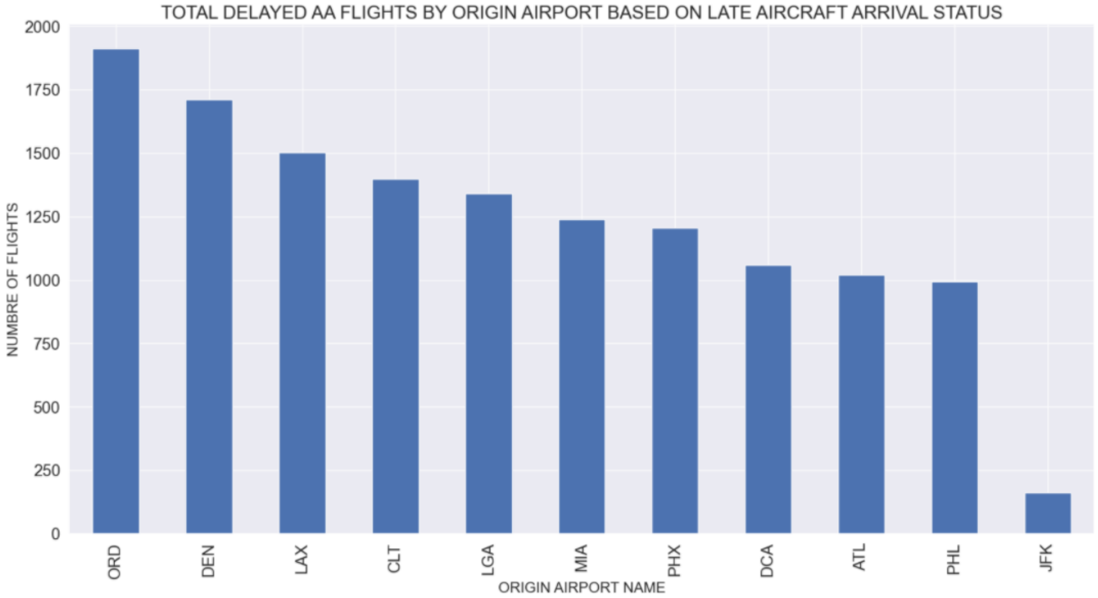
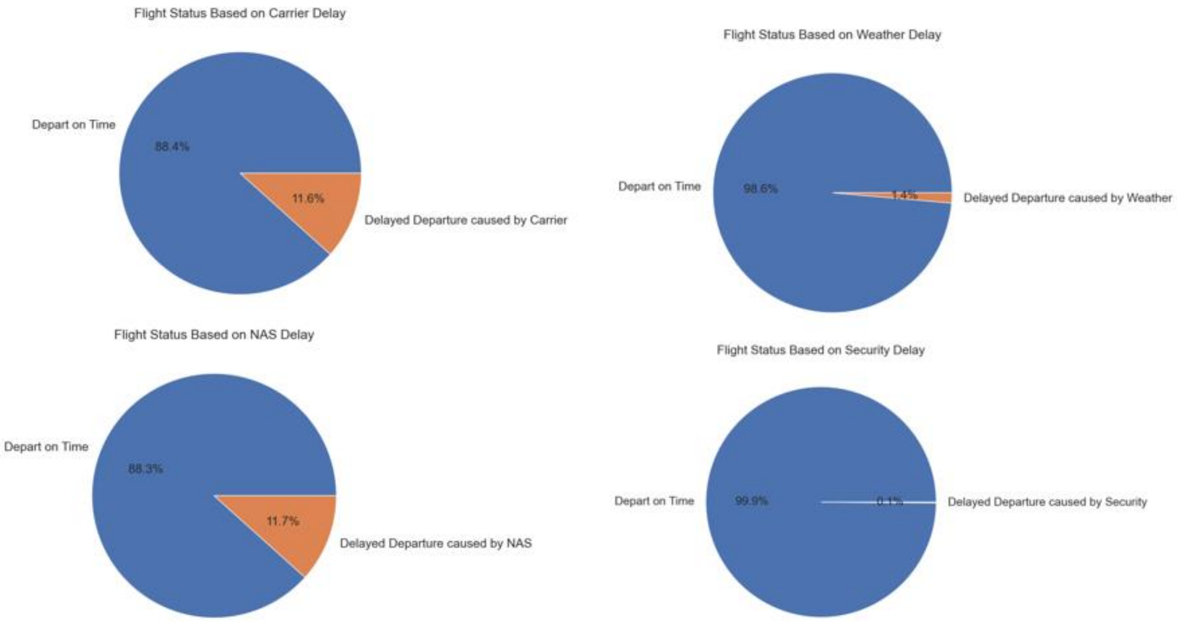


Figure 32: Total delayed AA flights by origin airport based on late air-craft delay

When observing all five causes of delay, it can be concluded that ORD is not great at handling any sort of delays except for security delays. With the help of data exploration, the **Error! Reference source not found.** pie chart shows what percentage contributes to each of the delays, such as 11.6% of flights are delayed by carrier delays, 1.4 % caused by weather delays, 11.7% are caused by NAS delays, 0.1% are caused by security delays and 9.0% are caused by late arrival delays.



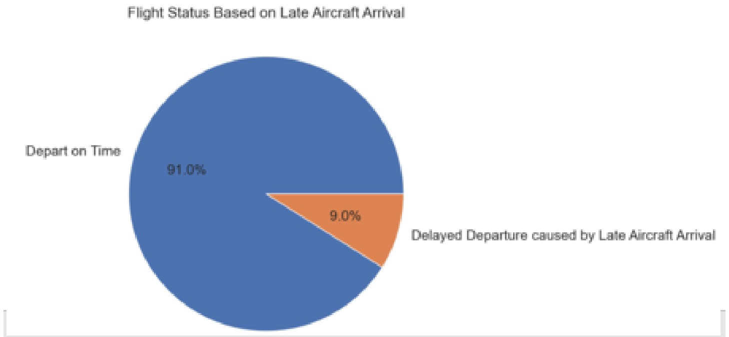


Figure 33:Percentage of flights delays based on carrier, weather, NAS, security, late aircraft

EDA: Exploratory Data Analysis for Southwest Airlines

Here, the data will be analyzed for Southwest Airlines arriving at DAL, this data was gathered to preprocess in the same fashion as American Airlines.

Using the dataset, which was limited to US domestic flights and the following airports: HOU, DEN, MDW, PHX, LAS, ATL, LAX, BWI, MCO, OAK, the objective is to find the total number of Southwest flights arrived at DAL during the year 2016-2019. The bar chart below shows the exact number of flights which arrived at DAL where the x-axis represents the origin airport and the y-axis represents the number of flights.

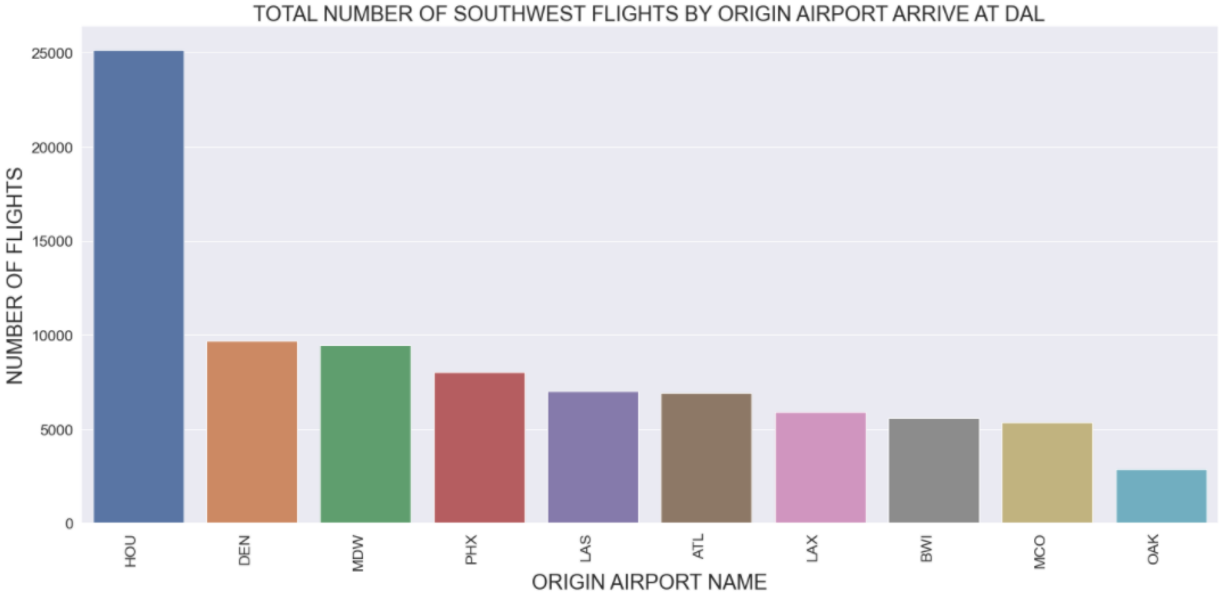


Figure 34: Total number of Southwest flights arrived at DAL

By using the same x and y-axis as in the above graph, and grouping origin and flight status in descending order, it can be observed that flights arriving from HOU to DAL have the highest number of delays. As one would think that the higher the number of flights arriving from a particular airport has a higher number of possibilities of being delayed, that is exactly the case when the below bar chart is observed.

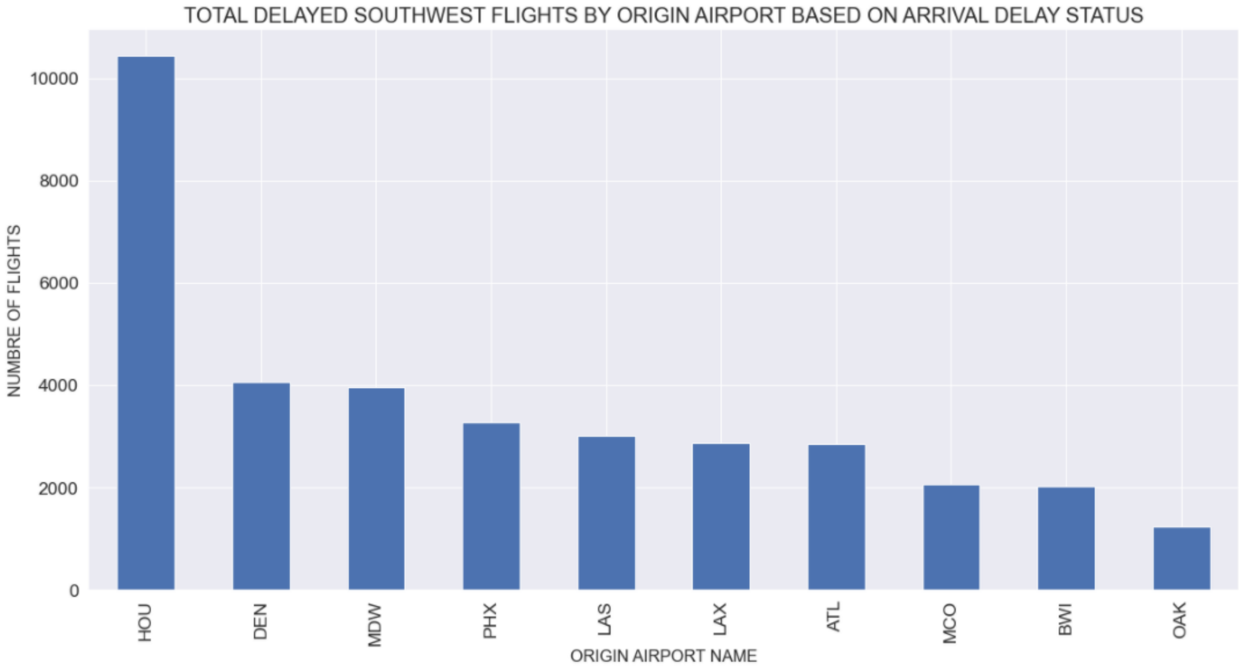


Figure 35: Total delayed SW flights based on arrival delay

As seen for AA there is a relationship between the departure flight status and the arrival flight status. Similarly, this pattern is found in Southwest Airlines as well. The Arrival Flight status shows that 58.4% of flights from the dataset have arrived on time and 41.6% were delayed. The Departure Flight Status, illustrates that 52.1 % of flights departed on time and 47.9% of flights departed late shown below.

COMPARISON BETWEEN DEPARTURE AND ARRIVAL DELAY (DAL)

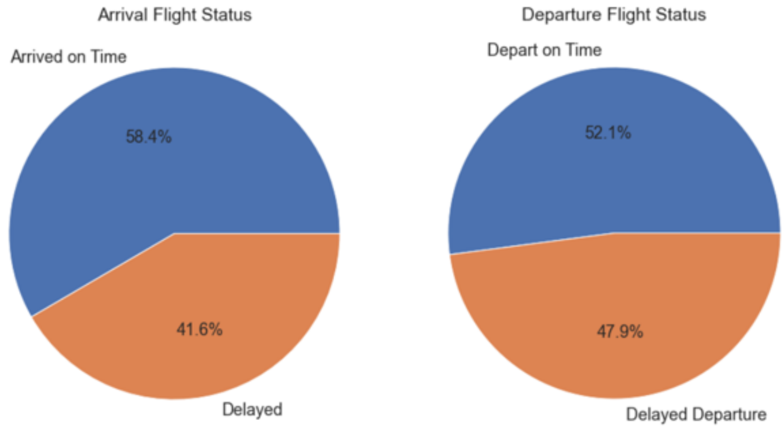


Figure 36:Comparison between departure and arrival delay

Among the five causes of delay, it is shown below that HOU deals with all delays quite poorly and OAK has the least number of overall delays, a reason as to why it has the least number of overall delays may be fact because it has the least number of flights coming from OAK.

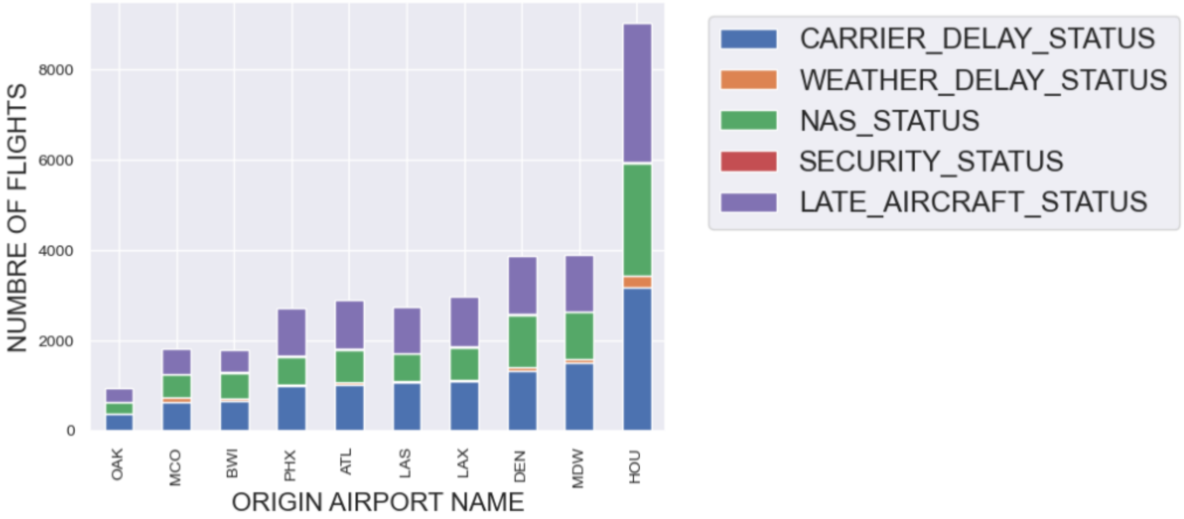


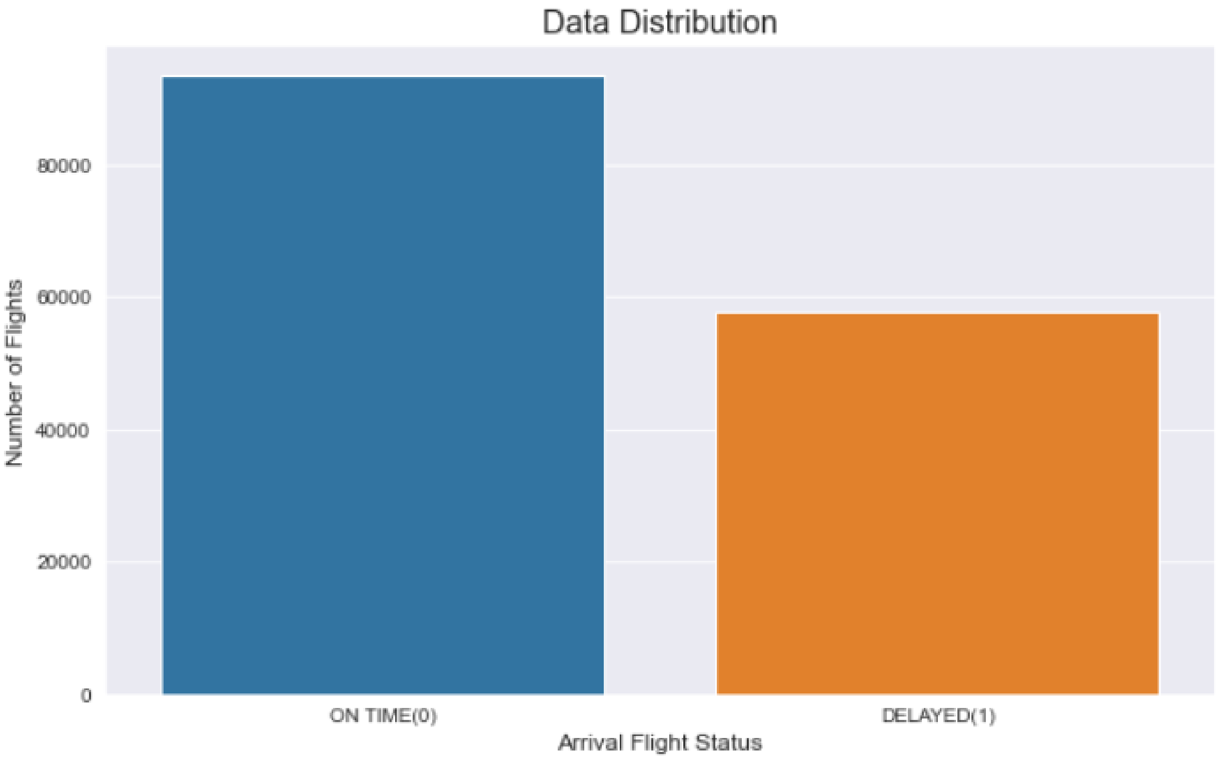
Figure 37: Total delayed SW flights by origin based on cause of delay

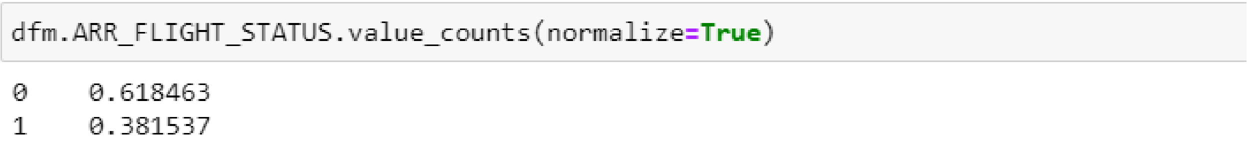
Chapter 4

RESULT AND DISCUSSION

MODELS using Python for American Airlines

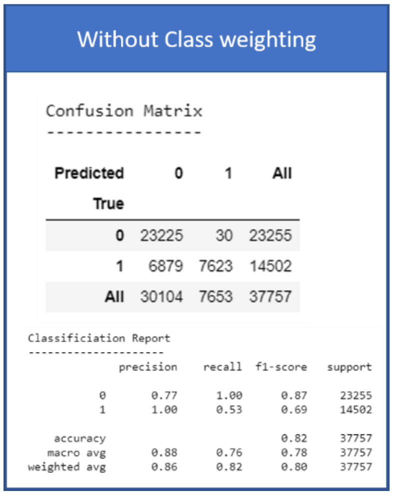
For this thesis, flight dataset from BTS from 2016 to 2019 will be used to understand class imbalance for flight arrival status. Figure below shows the distribution of the flight status. Based on the distribution for both AA-DFW and WN-DAL ration of on time vs delayed flight is less than 2:1. So the data is not highly imbalanced.





```
dfm.ARR_FLIGHT_STATUS.value_counts(normalize=True)

0    0.618463
1    0.381537
```

Random forest Model for American Airlines was utilized without and with different class weights (e.g., balanced and subsample balanced), to check the effect of the data imbalance on the performance matrices of the model. Below figure is a comparison of that result. As the data is not highly imbalanced, performance matrices do not vary much without or with class weights.

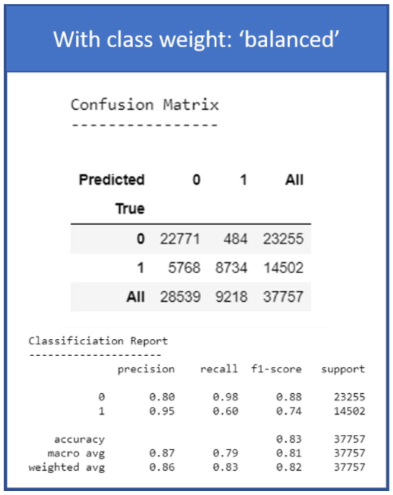


Without Class weighting

Confusion Matrix

Predicted	0	1	All
True			
0	23225	30	23255
1	6879	7623	14502
All	30104	7653	37757

```
Classificiation Report
--------------------
              precision    recall  f1-score   support

           0       0.77      1.00      0.87     23255
           1       1.00      0.53      0.69     14502

    accuracy                           0.82     37757
   macro avg       0.88      0.76      0.78     37757
weighted avg       0.86      0.82      0.80     37757
```

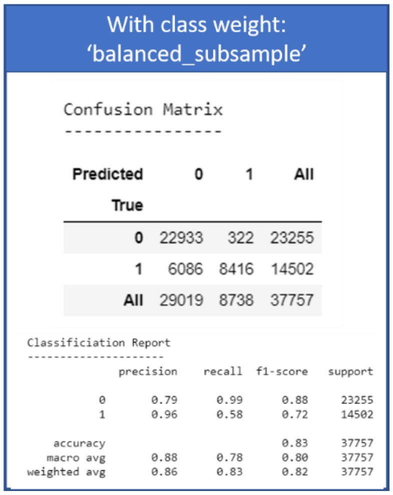


With class weight: ‘balanced’

Confusion Matrix

Predicted	0	1	All
True			
0	22771	484	23255
1	5768	8734	14502
All	28539	9218	37757

```
Classificiation Report
--------------------
              precision    recall  f1-score   support

           0       0.80      0.98      0.88     23255
           1       0.95      0.60      0.74     14502

    accuracy                           0.83     37757
   macro avg       0.87      0.79      0.81     37757
weighted avg       0.86      0.83      0.82     37757
```



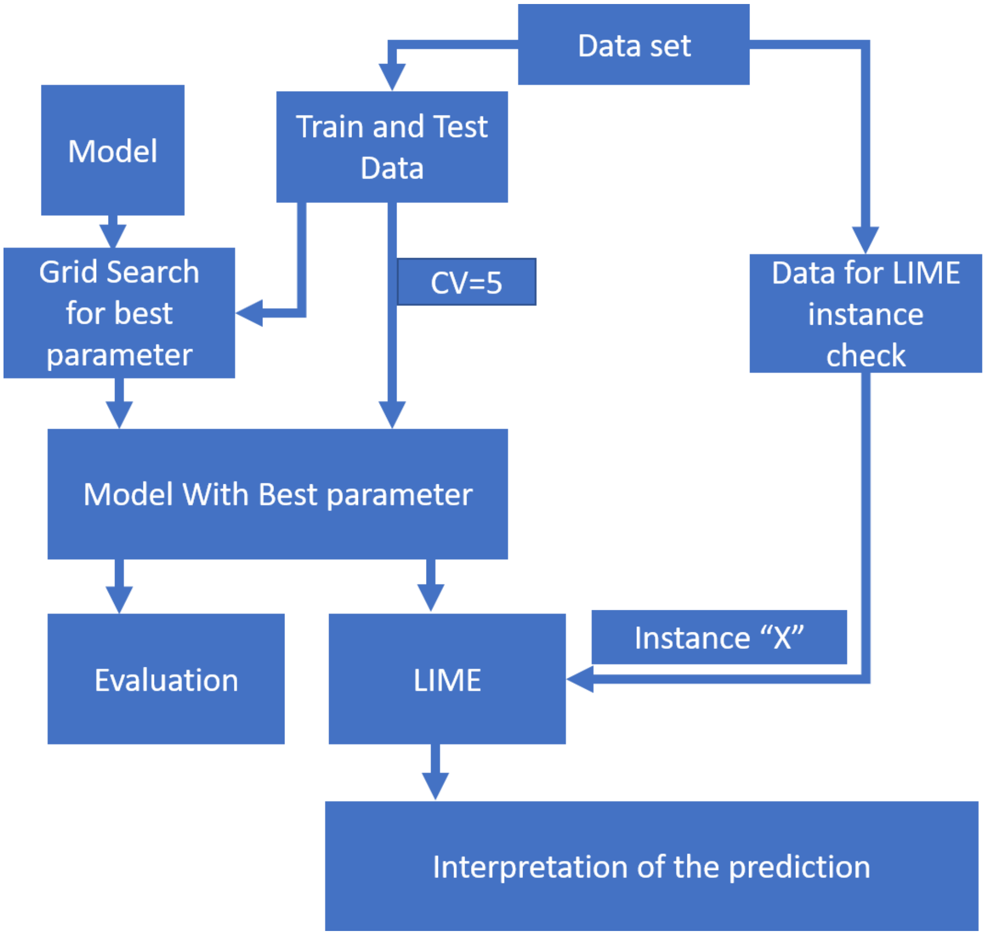
With class weight: ‘balanced_subsample’

Confusion Matrix

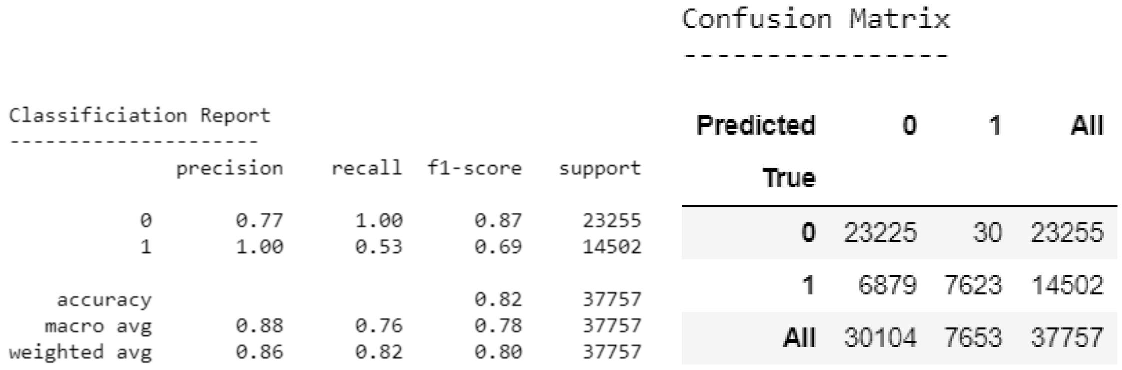
Predicted	0	1	All
True			
0	22933	322	23255
1	6086	8416	14502
All	29019	8738	37757

```
Classificiation Report
--------------------
              precision    recall  f1-score   support

           0       0.79      0.99      0.88     23255
           1       0.96      0.58      0.72     14502

    accuracy                           0.83     37757
   macro avg       0.88      0.78      0.80     37757
weighted avg       0.86      0.83      0.82     37757
```

The figure below represents the flow process to get the evaluation (performance matrices) and interpretation using LIME for all the models.


Data set
Model
Train and Test Data
Grid Search for best parameter
CV=5
Data for LIME instance check
Model With Best parameter
Evaluation
LIME
Instance “X”
Interpretation of the prediction

We already observed that data set is not highly imbalance, so the models were build and evaluated without any treatment for data imbalance. Below are the performance matrices and confusion matrix for Random forest. Accuracy of the model is 82%.



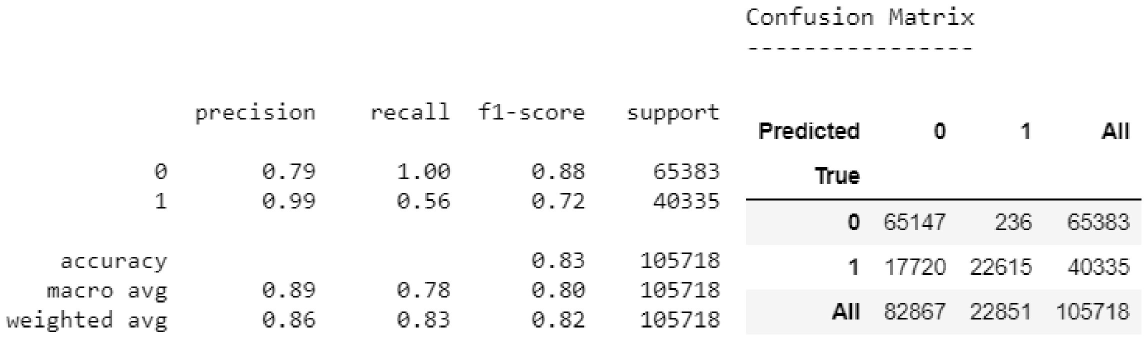
```
Classificiation Report
--------------------
              precision    recall  f1-score   support

           0       0.77      1.00      0.87     23255
           1       1.00      0.53      0.69     14502

    accuracy                           0.82     37757
   macro avg       0.88      0.76      0.78     37757
weighted avg       0.86      0.82      0.80     37757
```

```
Confusion Matrix
----------------
```

Predicted	0	1	All
True			
0	23225	30	23255
1	6879	7623	14502
All	30104	7653	37757

Figure 38:Evaluation Metric for random forest

Accuracy of Adaboost is 86%, Figure 39 shows the evaluation metrics for Adaboost machine learning model.



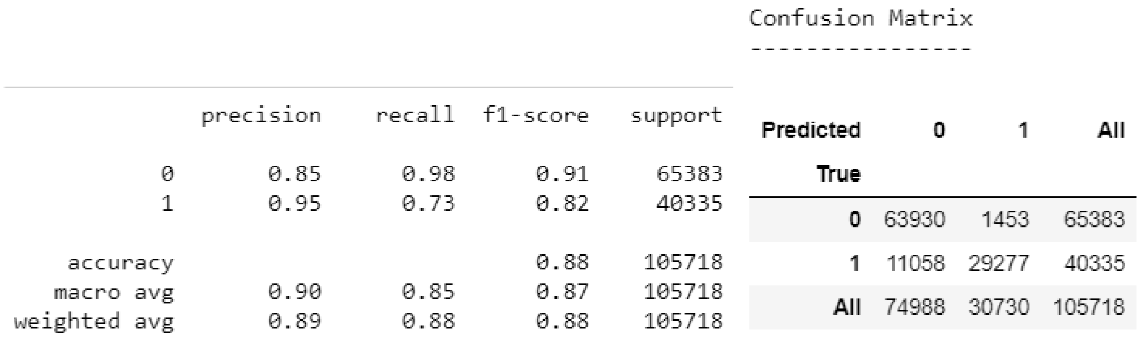
```
              precision    recall  f1-score   support

           0       0.79      1.00      0.88     65383
           1       0.99      0.56      0.72     40335

    accuracy                           0.83    105718
   macro avg       0.89      0.78      0.80    105718
weighted avg       0.86      0.83      0.82    105718
```

```
Confusion Matrix
----------------
```

Predicted	0	1	All
True			
0	65147	236	65383
1	17720	22615	40335
All	82867	22851	105718

Figure 39:Evaluation Metric for Adaboost

GradientBoost yields an accuracy of 88%. Below figure shows other performance matrices of GradientBoost.

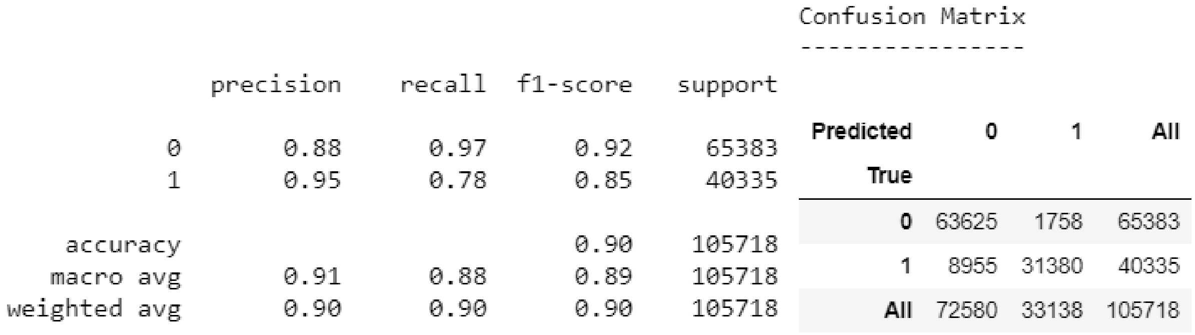


```
              precision    recall  f1-score   support

           0       0.85      0.98      0.91     65383
           1       0.95      0.73      0.82     40335

    accuracy                           0.88    105718
   macro avg       0.90      0.85      0.87    105718
weighted avg       0.89      0.88      0.88    105718
```

```
Confusion Matrix
----------------
```

Predicted	0	1	All
True			
0	63930	1453	65383
1	11058	29277	40335
All	74988	30730	105718

The XGBoost algorithm is highly flexible and faster in comparison to other algorithms, but it has several hyperparameters, many of which require tuning to obtain accurate results. One of the hyperparameters which is used is the scale_pos_weight hyperparameter which tunes imbalanced data. Using cross validation testing and training, the method obtained 91% accuracy shown on Figure 40.



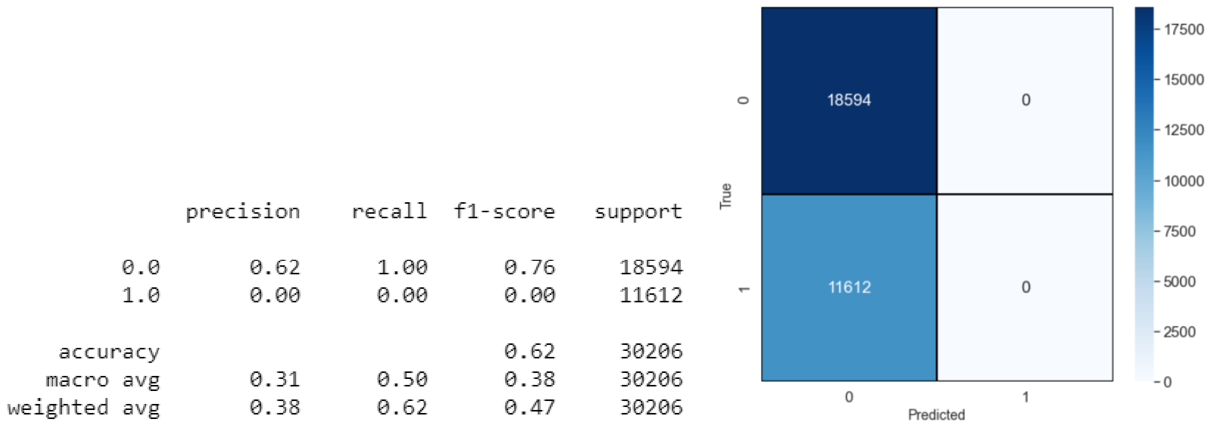
```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92     65383
           1       0.95      0.78      0.85     40335

    accuracy                           0.90    105718
   macro avg       0.91      0.88      0.89    105718
weighted avg       0.90      0.90      0.90    105718
```

```
Confusion Matrix
----------------
```

Predicted	0	1	All
True			
0	63625	1758	65383
1	8955	31380	40335
All	72580	33138	105718

Figure 40: Evaluation Metric for XGBoost

When using multilayered perceptron – neural network (MLP-NN), the data was required to be entirely numerical as it does not function with qualitative data, similarly it does not function with data that contains missing values. Using the StandardScale function the data was standardized and below is the output from the confusion matrix, which obtained an accuracy of 62%, shown in Figure 41.



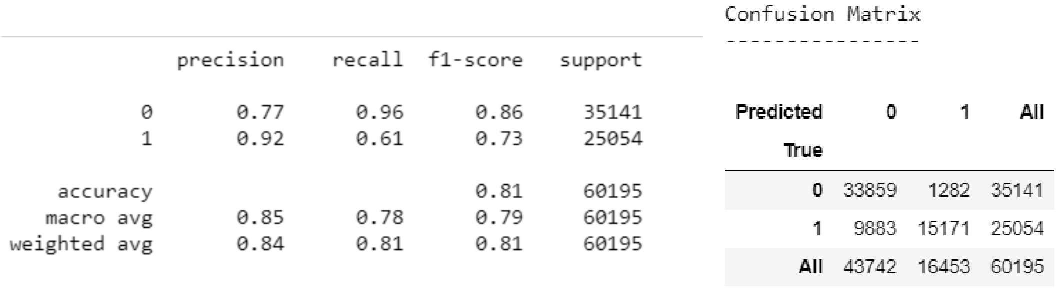
```
              precision    recall  f1-score   support

         0.0       0.62      1.00      0.76     18594
         1.0       0.00      0.00      0.00     11612

    accuracy                           0.62     30206
   macro avg       0.31      0.50      0.38     30206
weighted avg       0.38      0.62      0.47     30206
```

Figure 41: Evaluation Metric for MLP-NN

MODELS using Python for Southwest Airlines

In the above section for Models using Python and Results for AA were shared, the same procedure was applied to the Southwest Airlines dataset and the results for Random Forest shown on Figure 42, Adaboost, XGBoost, MLP-NN can be found below.

Random Forest



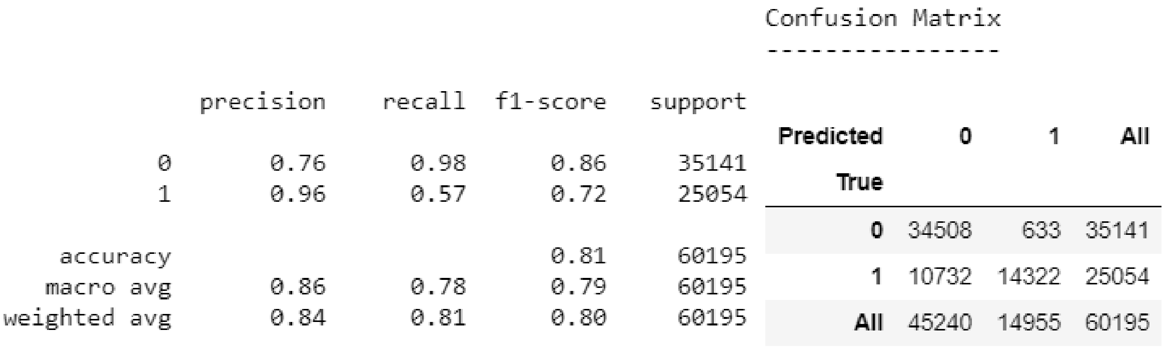
```
              precision    recall  f1-score   support

           0       0.77      0.96      0.86     35141
           1       0.92      0.61      0.73     25054

    accuracy                           0.81     60195
   macro avg       0.85      0.78      0.79     60195
weighted avg       0.84      0.81      0.81     60195
```

```
Confusion Matrix
----------------
```

Predicted True	0	1	All
0	33859	1282	35141
1	9883	15171	25054
All	43742	16453	60195

Figure 42:Evaluation Metric for Random Forest

Adaboost



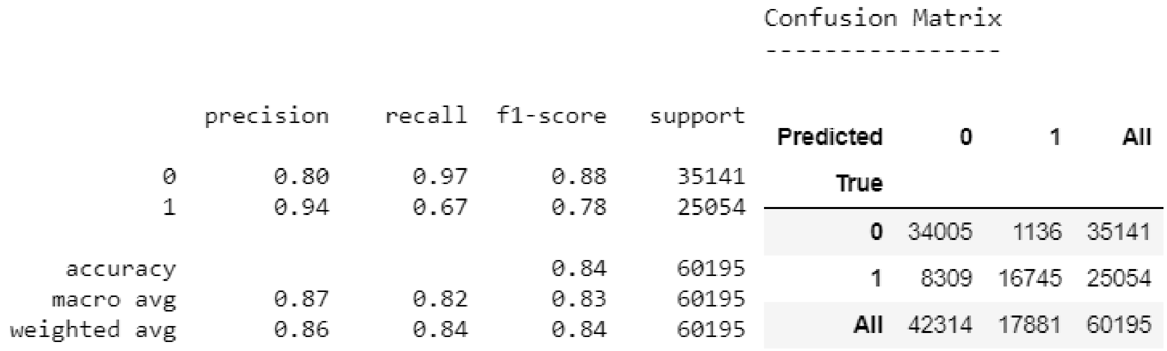
```
              precision    recall  f1-score   support

           0       0.76      0.98      0.86     35141
           1       0.96      0.57      0.72     25054

    accuracy                           0.81     60195
   macro avg       0.86      0.78      0.79     60195
weighted avg       0.84      0.81      0.80     60195
```

```
Confusion Matrix
----------------
```

Predicted True	0	1	All
0	34508	633	35141
1	10732	14322	25054
All	45240	14955	60195

Figure 43:Evaluation Metric for Adaboost

GradientBoost

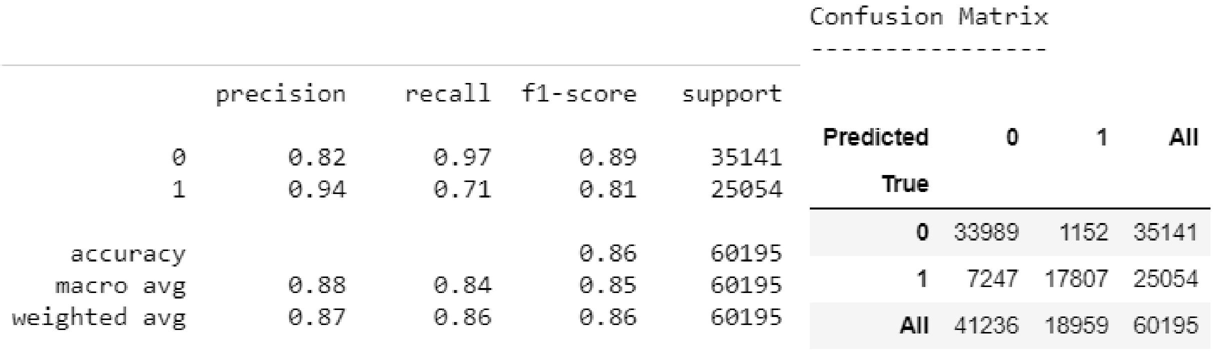


```
              precision    recall  f1-score   support

           0       0.80      0.97      0.88     35141
           1       0.94      0.67      0.78     25054

    accuracy                           0.84     60195
   macro avg       0.87      0.82      0.83     60195
weighted avg       0.86      0.84      0.84     60195
```

```
Confusion Matrix
----------------
```

Predicted	0	1	All
True			
0	34005	1136	35141
1	8309	16745	25054
All	42314	17881	60195

XGBoost



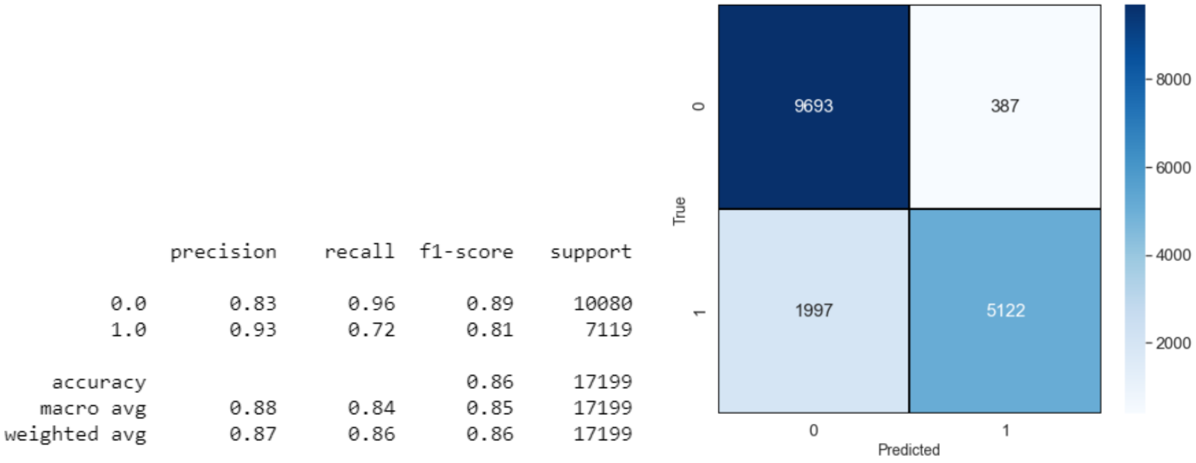
```
              precision    recall  f1-score   support

           0       0.82      0.97      0.89     35141
           1       0.94      0.71      0.81     25054

    accuracy                           0.86     60195
   macro avg       0.88      0.84      0.85     60195
weighted avg       0.87      0.86      0.86     60195
```

```
Confusion Matrix
----------------
```

Predicted	0	1	All
True			
0	33989	1152	35141
1	7247	17807	25054
All	41236	18959	60195

Figure 44:Evaluation Metric for xgboost

MLP-NN



```
              precision    recall  f1-score   support

         0.0       0.83      0.96      0.89     10080
         1.0       0.93      0.72      0.81      7119

    accuracy                           0.86     17199
   macro avg       0.88      0.84      0.85     17199
weighted avg       0.87      0.86      0.86     17199
```

Figure 45: Evaluation Metric for MLP- NN

Below tables have the performance metrics of the models

American Airline and DFW:

Table 2: Performance metrics for AA

Model Name	Accuracy (%)	Precision(%)	Recall(%)	F1-score(%)
Random Forest	85	86	85	85
Adaboost	83	86	83	82
Gradient Boost	88	89	88	88
XGBoost	90	90	90	90
MLP - NN	86	92	92	92

Southwest and DAL:

Table 3:: Performance metrics for Southwest

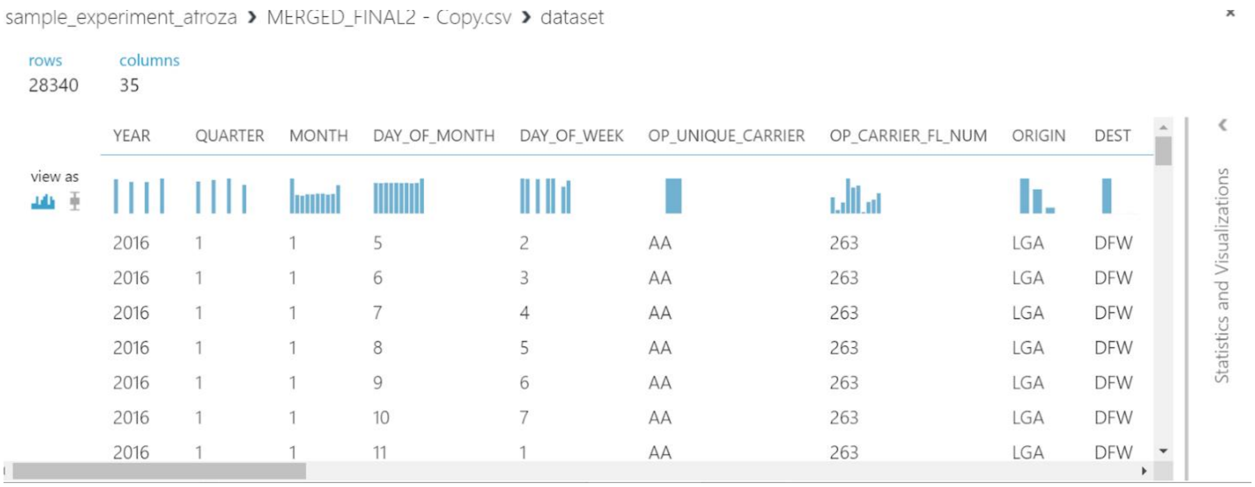
Model Name	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)
Random Forest	81	84	81	81
Adaboost	81	84	81	80
Gradient Boost	84	86	84	84
XGBoost	86	87	86	86
MLP - NN	86	87	86	86

Microsoft Azure Studio

Previously, the experiment was done using Python, the same dataset will be used with Azure Machine Learning studio to compare between the two. The studio offers no-code experiences for a data science platform and creates models for classification, regression and time-series forecasting depending on the type of project that is being worked on. In this experiment, the classification approach will be used, which is a supervised learning problem. It will be used to predict whether the flight will be delayed or not, using a two-class decision forest.

In the experiment, a model was trained using historic flight data (from 2016-2019). The features are labeled “1” if a flight was delayed, and labeled “0” if the flight was on time. The following steps were followed while building the experiment in Azure ML Studio: the data was imported, and pre-processed, the data was trained, and a machine learning algorithm was applied, followed by executing score, the model was then tested and the accuracy was predicted.

A dataset requires pre-processing before it can be analyzed, using the tools provided by the studio the following pre-processing changes were made.



sample_experiment_atroza › MERGED_FINAL2 - Copy.csv › dataset

rows 28340

columns 35

YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	OP_UNIQUE_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST
2016	1	1	5	2	AA	263	LGA	DFW
2016	1	1	6	3	AA	263	LGA	DFW
2016	1	1	7	4	AA	263	LGA	DFW
2016	1	1	8	5	AA	263	LGA	DFW
2016	1	1	9	6	AA	263	LGA	DFW
2016	1	1	10	7	AA	263	LGA	DFW
2016	1	1	11	1	AA	263	LGA	DFW

Figure 46: Visualization from Azure

Steps:

- Loaded the dataset
- Replaced “NaN” with 0 for feature 'CARRIER_DELAY', WEATHER_DELAY', 'NAS_DELAY','SECURITY_DELAY','LATE_AIRCRAFT_DELAY’
- Dropped feature 'DEP_TIME', 'ARR_TIME’ because of redundancy

- All the missing data for TAXI_IN, TAXI_OUT, DEP_DELAY, DEP_DELAY_NEW, ARR_DELAY, and ARR_DELAY_NEW are filled with the mean value of each feature

- For feature 'CRS_DEP_TIME’, 'WHEELS_OFF’, 'WHEELS_ON’, 'CRS_ARR_TIME’, Time is normally categorical, and having it in the current format will give us too many columns when the hot encode is applied to them, therefore it will be better to split the time into 4 quarters of the days meaning of 6 hours each

- For the DAY_OF_WEEK feature are present 0 = Monday, 1 = Tuesday, 2 = Wednesday, 3 = Thursday, 4 = Friday, 5 = Saturday, 6 = Sunday

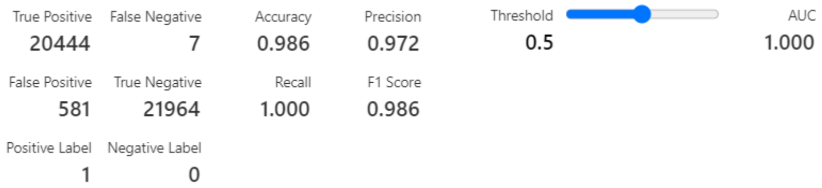
- The feature ARR_DELAY, reveals the difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers. Therefore, created a new Column ARR_FLIGHT_STATUS, by comparing the value of each row for the "ARR_DELAY" feature, if the total delay zero or less (minus-- arrive early) it will assign "0" otherwise "1".

- Similarly, for DEP_DELAY, CARRIER_DELAY, 'WEATHER_DELAY’, 'NAS_DELAY’, 'SECURITY_DELAY’, 'LATE_AIRCRAFT_DELAY’, created, DEP_FLIGHT_STATUS, CARRIER_DELAY_STATUS, WEATHER_DELAY_STATUS,

- NAS_STATUS,SECURITY_STATUS,LATE_AIRCRAFT_STATUS

- There are some feature which are duplicate in nature: ARR_DELAY, DEP_DELAY, CARRIER_DELAY,WEATHER_DELAY, NAS_DELAY,SECURITY_DELAY , LATE_AIRCRAFT_STATUS, 'DEP_DELAY_NEW', 'DEP_DEL15','ARR_DELAY_NEW', 'ARR_DEL15','CANCELLED','CANCELLATION_CODE', 'DIVERTED’

For comparison with Python, the model was created using the Two Class Decision Forest ,which was very close to the random forest model applied in Python. The result of the experiment is shown below.



True Positive 20444	False Negative 7	Accuracy 0.986	Precision 0.972	Threshold 0.5	AUC 1.000
False Positive 581	True Negative 21964	Recall 1.000	F1 Score 0.986		
Positive Label 1	Negative Label 0				

Figure 47: Result for Two-Class Decision Forest

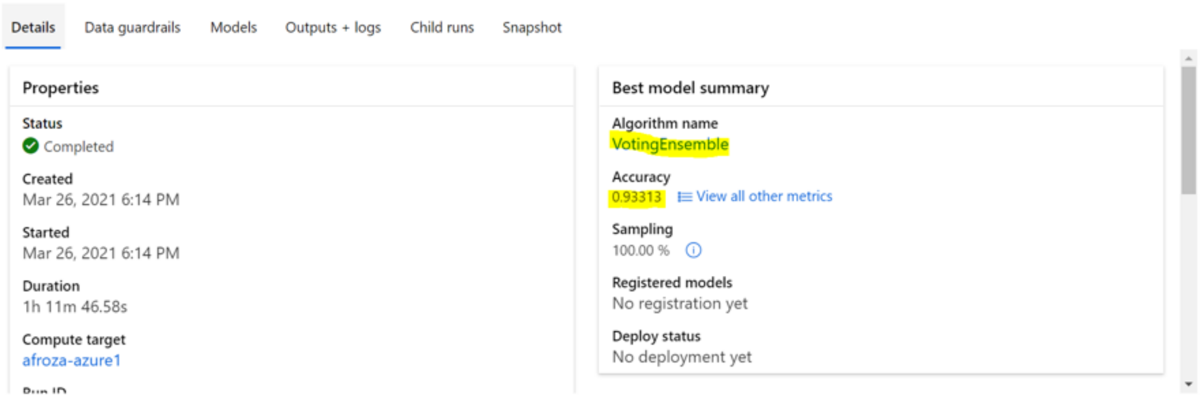
Microsoft Azure Auto-ML

Automated machine learning (AutoML) automates the process of building machine learning models through automating the feature selection and parametrization of these models. AutoML has been designed to helps data scientists perform this process with accuracy and precision. Azure Machine Learning offers featurization for classification problems. Classification models predict from the provided training data. When the data is being trained, AutoML will create various pipelines using multiple algorithms. The system will then, iterate through these algorithms alongside feature selections and will produce a model as output from the training dataIdentify the ML problem: classification, or regression

1. Choose no code experience in Azure Machine Learning studio
2. Specify the source and format of the labeled training data
3. Configure the compute target for model training.
4. Configure the automated machine learning parameters such as how many iterations there are in the models and tune hyperparameters.
5. Submit the training data.

6. Review the results

This experiment created a classification model, without writing code using automated machine learning in the Azure Machine Learning studio, the same dataset was used for comparison. The studio also provided model explanations which allowed for the observation of which data features influenced a particular model's predictions. The accuracy obtained by Azure- AutoML when using XGBoost was 82%, and 92% when using Random Forest . Although the best Algorithm suggested by Azure- AutoML is VotingEnsemble. Although in this thesis VotingEnsemble was not used in Python, it combines the predictions from multiple other models, it helps improve model performance.



Properties

Status
Completed

Created
Mar 26, 2021 6:14 PM

Started
Mar 26, 2021 6:14 PM

Duration
1h 11m 46.58s

Compute target
afroza-azure1

Run ID

Best model summary

Algorithm name
VotingEnsemble

Accuracy
0.93313 View all other metrics

Sampling
100.00 %

Registered models
No registration yet

Deploy status
No deployment yet

Figure 48:Result for Azure Auto-ML

Interpretability of the models prediction using LIME:

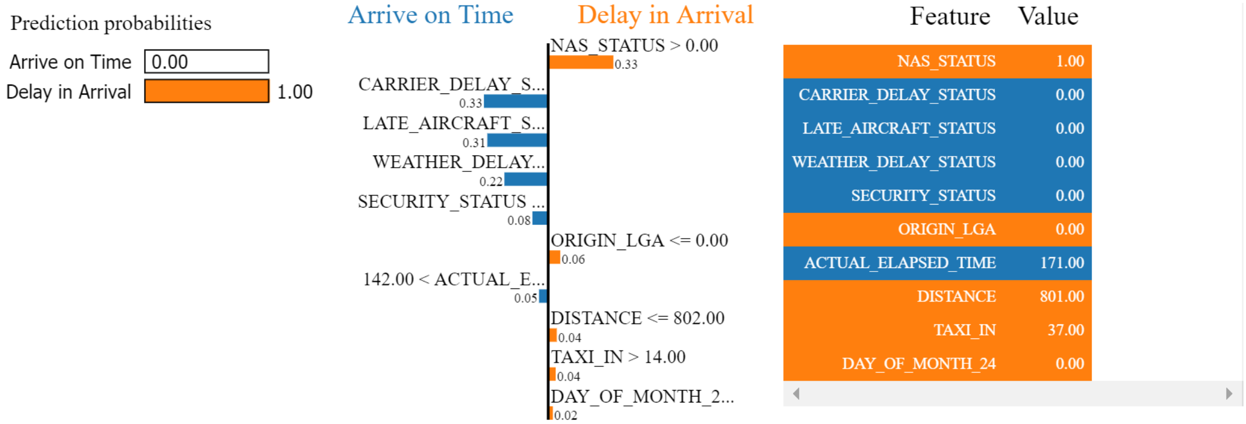
To use LIME algorithm in Python, the library “lime” needed to be installed. Randomly two flights were chosen from the test data set to explain the use of LIME.

All Model Comparison for LIME for a specific Flight from test data set

Case-1: Delayed Flight(AA)

All model predicted that this flight will be delayed with different confidence level. Random forest and XGBoost predicted the delay with 100% confidence but Adaboost’s predict the delay for this flight with 68% probability. All model selected NAS delay as the top feature for the delay of this flight.

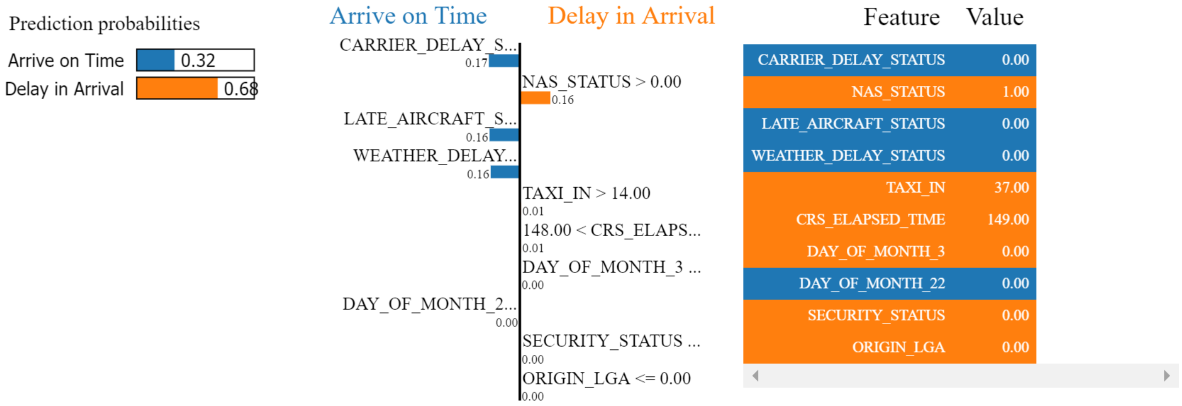
Random Forest:



Feature	Value
NAS_STATUS	1.00
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
SECURITY_STATUS	0.00
ORIGIN_LGA	0.00
ACTUAL_ELAPSED_TIME	171.00
DISTANCE	801.00
TAXI_IN	37.00
DAY_OF_MONTH_24	0.00

Figure 49: Case 1 result using Random Forest

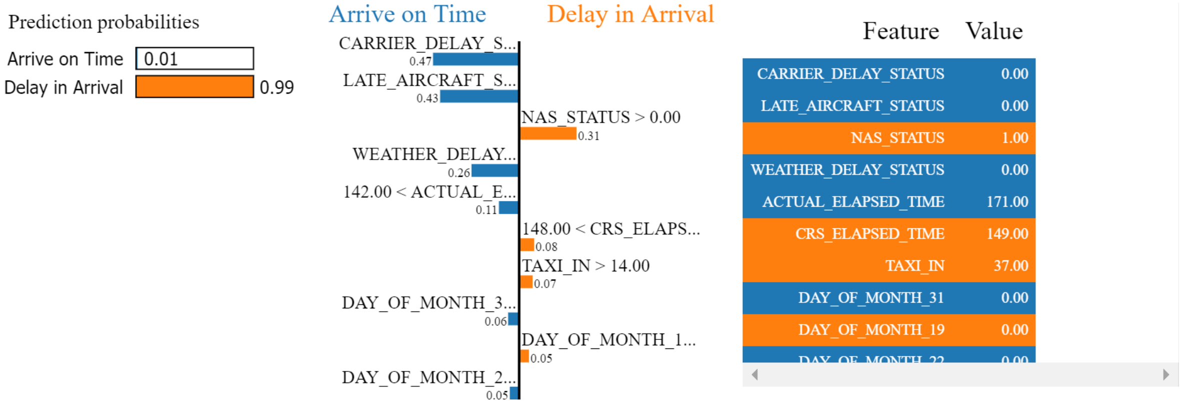
Adaboost:



Feature	Value
CARRIER_DELAY_STATUS	0.00
NAS_STATUS	1.00
LATE_AIRCRAFT_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
TAXI_IN	37.00
CRS_ELAPSED_TIME	149.00
DAY_OF_MONTH_3	0.00
DAY_OF_MONTH_22	0.00
SECURITY_STATUS	0.00
ORIGIN_LGA	0.00

Figure 50: Case 1 result using Adaboost

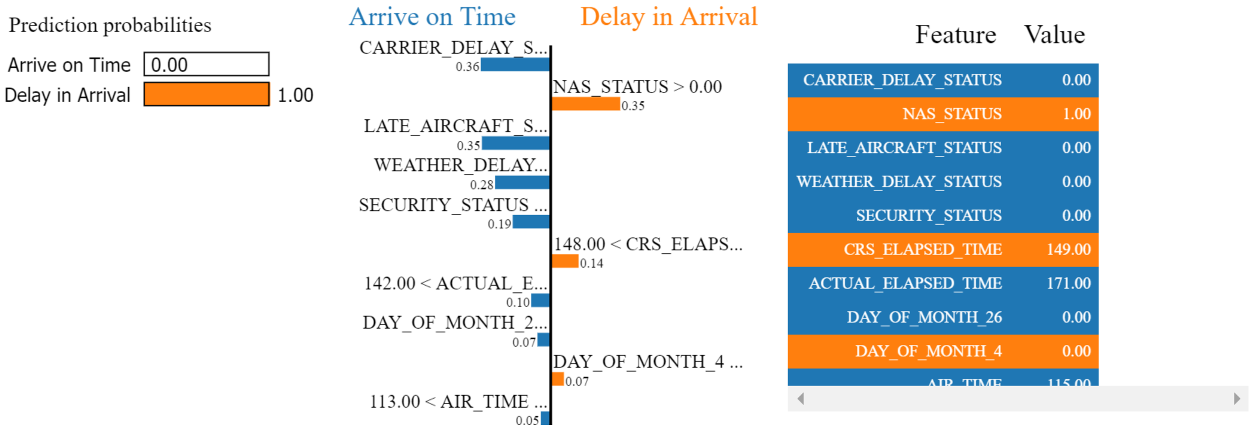
Gradient Boosting:



Feature	Value
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
NAS_STATUS	1.00
WEATHER_DELAY_STATUS	0.00
ACTUAL_ELAPSED_TIME	171.00
CRS_ELAPSED_TIME	149.00
TAXI_IN	37.00
DAY_OF_MONTH_31	0.00
DAY_OF_MONTH_19	0.00
DAY_OF_MONTH_22	0.00

Figure 51: Case 1 result using Gradient Boosting

XGBoost:



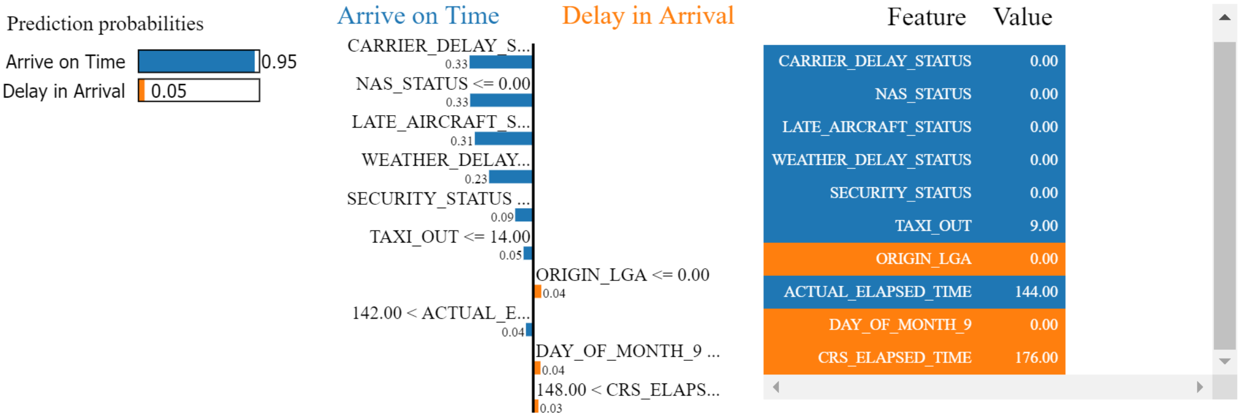
Feature	Value
CARRIER_DELAY_STATUS	0.00
NAS_STATUS	1.00
LATE_AIRCRAFT_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
SECURITY_STATUS	0.00
CRS_ELAPSED_TIME	149.00
ACTUAL_ELAPSED_TIME	171.00
DAY_OF_MONTH_26	0.00
DAY_OF_MONTH_4	0.00
AIR_TIME	115.00

Figure 52 : Case 1 result using xgboost

Case-2: Arrive on time Flight(AA)

In this case Random forest, XGBoost and Gradient boost predicted with above 90% probability that this flight arrives on time but Adaboost again showed poor prediction probabilities. As Carrier, NAS, Late Arrival, Weather and Security delay are not present there, most of the model pick those as most important features for the prediction.

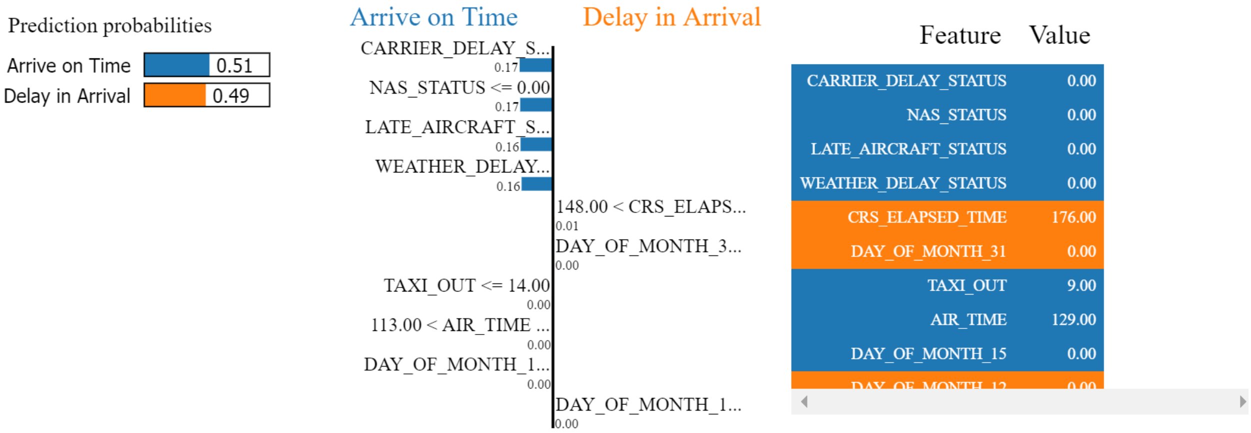
Random Forest:



Feature	Value
CARRIER_DELAY_STATUS	0.00
NAS_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
SECURITY_STATUS	0.00
TAXI_OUT	9.00
ORIGIN_LGA	0.00
ACTUAL_ELAPSED_TIME	144.00
DAY_OF_MONTH_9	0.00
CRS_ELAPSED_TIME	176.00

Figure 53 : Case 2 result using Random Forest

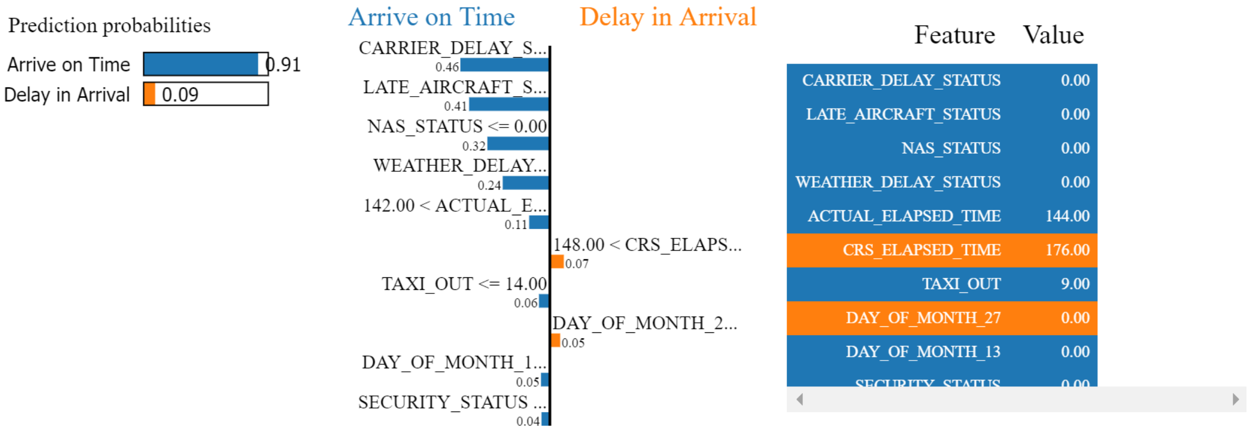
Adaboost:



Feature	Value
CARRIER_DELAY_STATUS	0.00
NAS_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
CRS_ELAPSED_TIME	176.00
DAY_OF_MONTH_31	0.00
TAXI_OUT	9.00
AIR_TIME	129.00
DAY_OF_MONTH_15	0.00
DAY_OF_MONTH_12	0.00

Figure 54 : Case 2 result using Adaboost

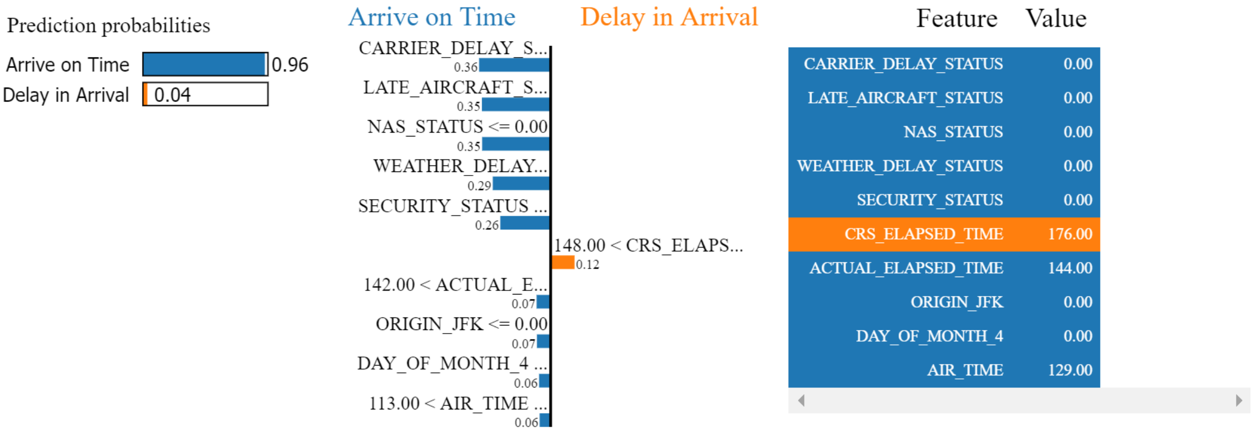
Gradient Boosting:



Feature	Value
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
NAS_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
ACTUAL_ELAPSED_TIME	144.00
CRS_ELAPSED_TIME	176.00
TAXI_OUT	9.00
DAY_OF_MONTH_27	0.00
DAY_OF_MONTH_13	0.00
SECURITY_STATUS	0.00

Figure 55 : Case 2 result using Gradient Boosting

XGBoost:



Feature	Value
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
NAS_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
SECURITY_STATUS	0.00
CRS_ELAPSED_TIME	176.00
ACTUAL_ELAPSED_TIME	144.00
ORIGIN_JFK	0.00
DAY_OF_MONTH_4	0.00
AIR_TIME	129.00

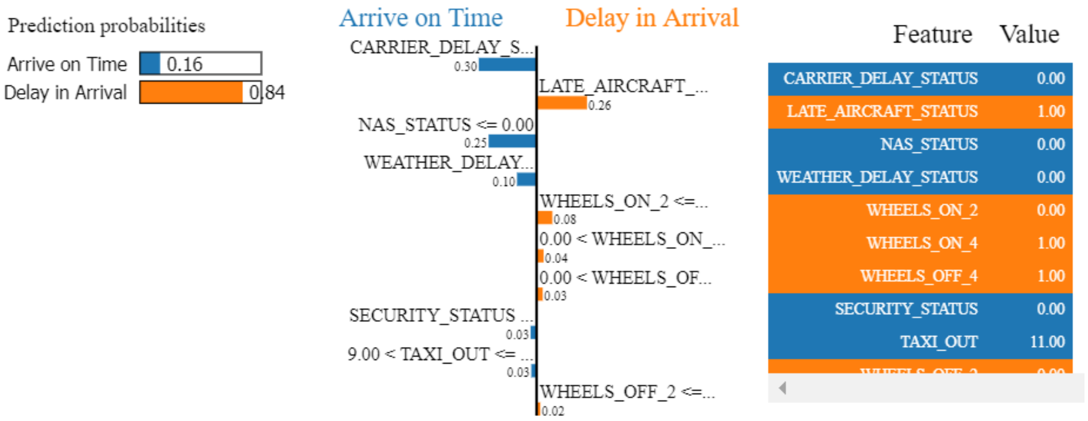
Figure 56: Case 2 result using XGBoosting

So, lime can be used to identify which model is suitable to make the accurate prediction with reasonable feature importance.

Case-1: Delayed Flight (Southwest)

All models predicted that this flight will be delayed with different confidence level. Random forest predicted 84% and GradientBoost predicted the delay with confidence 95%, XGBoost predicted 99% but Adaboost's predicted the delay for this flight with 65% probability.

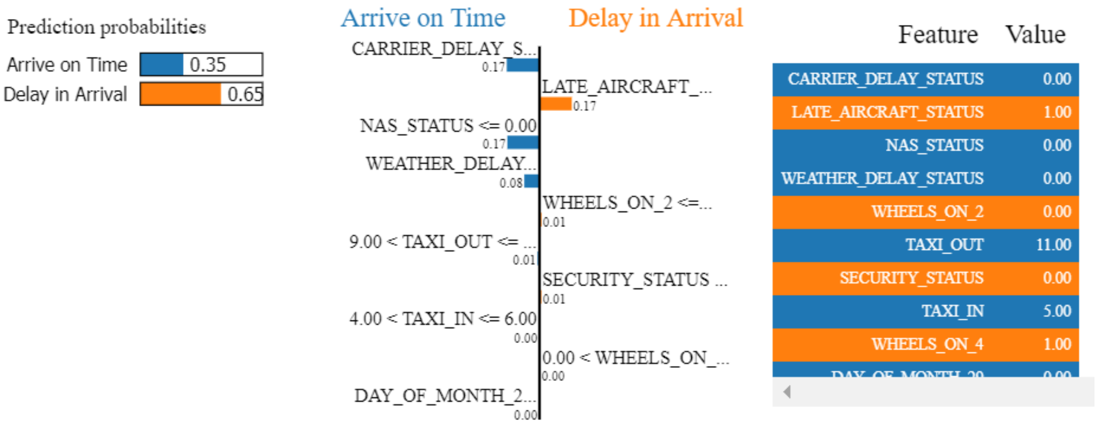
Random Forest:



Feature	Value
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	1.00
NAS_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
WHEELS_ON_2	0.00
WHEELS_ON_4	1.00
WHEELS_OFF_4	1.00
SECURITY_STATUS	0.00
TAXI_OUT	11.00
WHEELS_OFF_2	0.00

Figure 57:Case 1 result using Random Forest

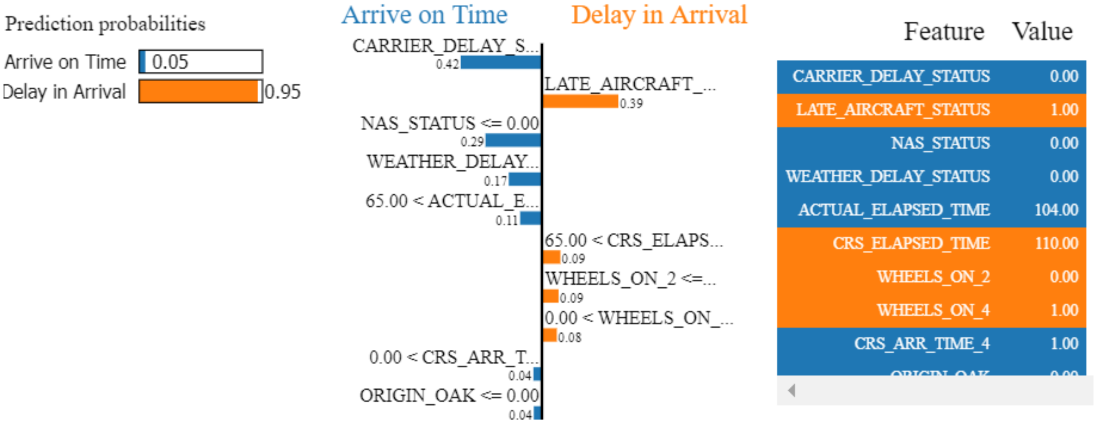
Adaboost:



Feature	Value
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	1.00
NAS_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
WHEELS_ON_2	0.00
TAXI_OUT	11.00
SECURITY_STATUS	0.00
TAXI_IN	5.00
WHEELS_ON_4	1.00
DAY_OF_MONTH_20	0.00

Figure 58:Case 1 result using Adaboost

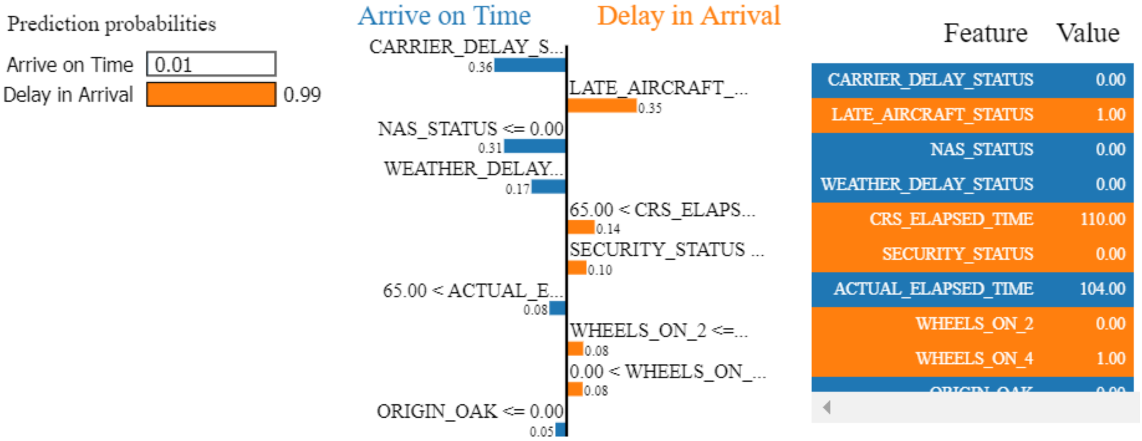
Gradient Boost:



Feature	Value
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	1.00
NAS_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
ACTUAL_ELAPSED_TIME	104.00
CRS_ELAPSED_TIME	110.00
WHEELS_ON_2	0.00
WHEELS_ON_4	1.00
CRS_ARR_TIME_4	1.00
ORIGIN_OAK	0.00

Figure 59:Case 1 result using Gradient Boost

XGBoost:



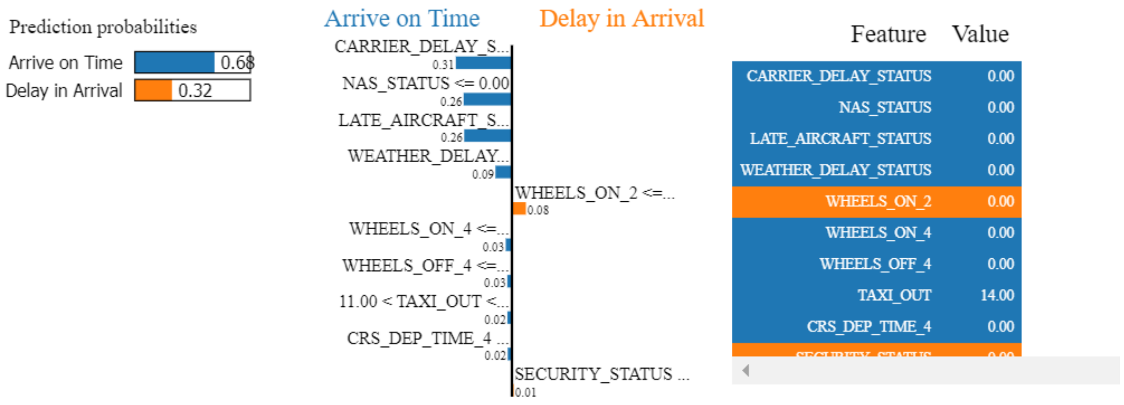
Feature	Value
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	1.00
NAS_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
CRS_ELAPSED_TIME	110.00
SECURITY_STATUS	0.00
ACTUAL_ELAPSED_TIME	104.00
WHEELS_ON_2	0.00
WHEELS_ON_4	1.00
ORIGIN_OAK	0.00

Figure 60:Case 1 result using XGBoost

Case-2: Arrive on time Flight(Southwest)

In this case Random forest, XGBoost and Gradient boost predicted with above 60% probability that this flight arrives on time but Adaboost again showed poor prediction probabilities.

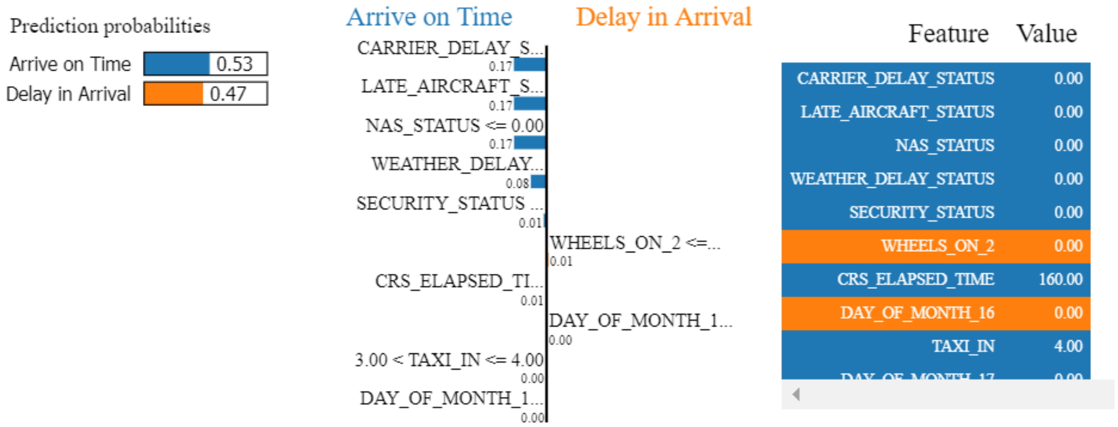
Random Forest:



Feature	Value
CARRIER_DELAY_STATUS	0.00
NAS_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
WHEELS_ON_2	0.00
WHEELS_ON_4	0.00
WHEELS_OFF_4	0.00
TAXI_OUT	14.00
CRS_DEP_TIME_4	0.00
SECURITY_STATUS	0.00

Figure 61:Case 2 result using Random Forest

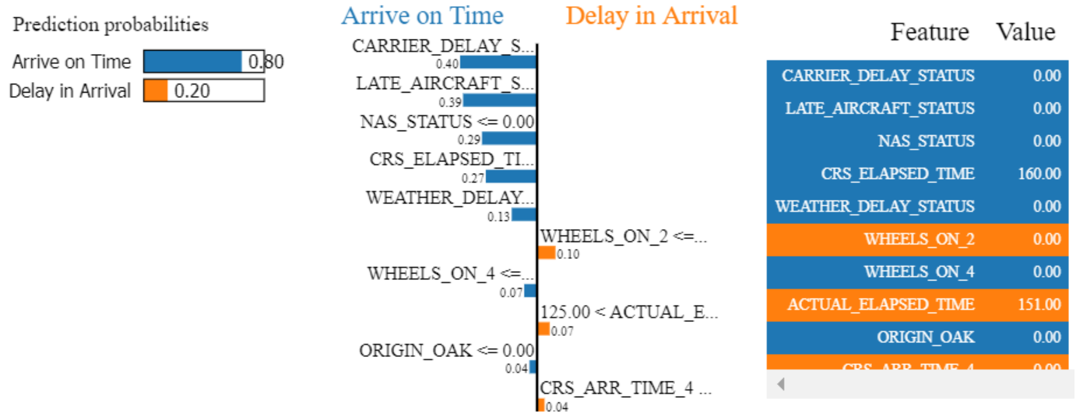
Adaboost:



Feature	Value
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
NAS_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
SECURITY_STATUS	0.00
WHEELS_ON_2	0.00
CRS_ELAPSED_TIME	160.00
DAY_OF_MONTH_16	0.00
TAXI_IN	4.00
DAY_OF_MONTH_17	0.00

Figure 62:Case 2 result using Adaboost

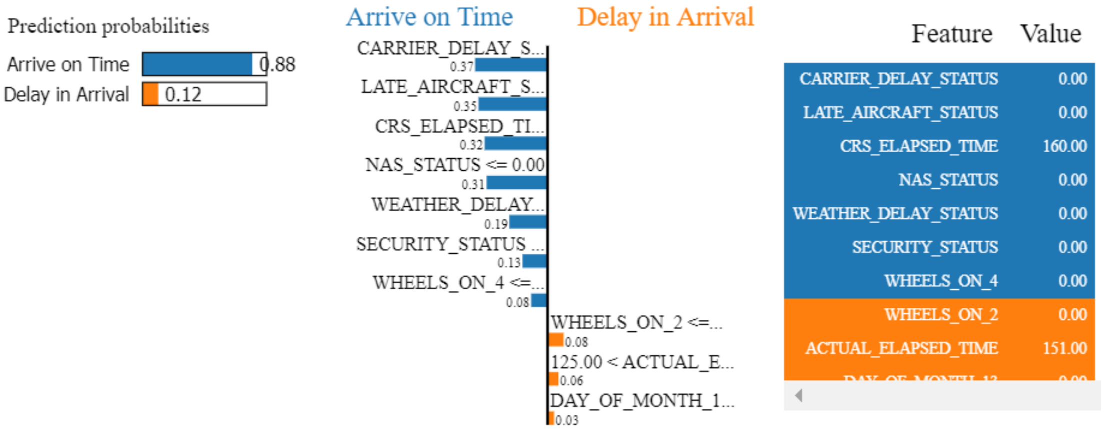
Gradient Boost:



Feature	Value
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
NAS_STATUS	0.00
CRS_ELAPSED_TIME	160.00
WEATHER_DELAY_STATUS	0.00
WHEELS_ON_2	0.00
WHEELS_ON_4	0.00
ACTUAL_ELAPSED_TIME	151.00
ORIGIN_OAK	0.00
CRS_ARR_TIME_4	0.00

Figure 63:Case 2 result using Gradient Boost

XGBoost:

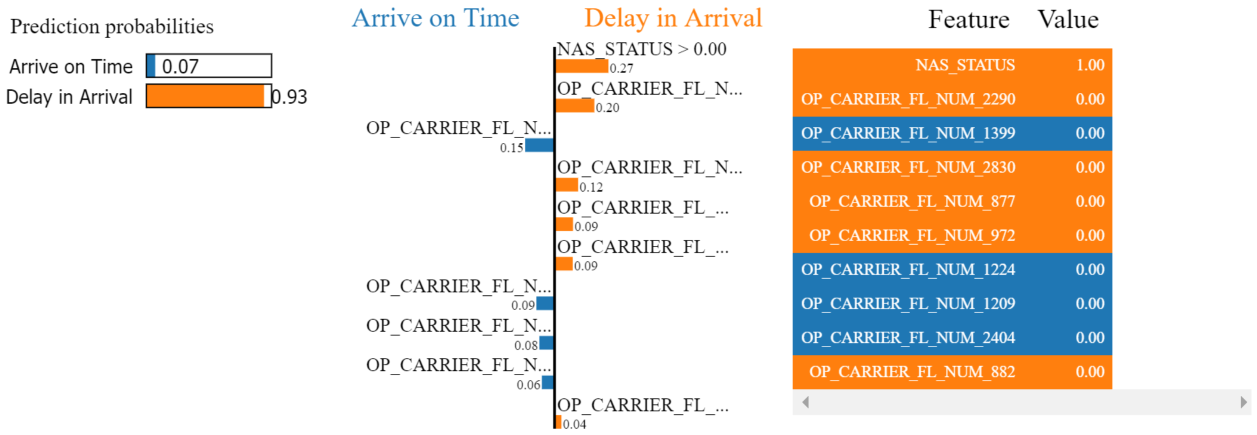


Feature	Value
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
CRS_ELAPSED_TIME	160.00
NAS_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
SECURITY_STATUS	0.00
WHEELS_ON_4	0.00
WHEELS_ON_2	0.00
ACTUAL_ELAPSED_TIME	151.00
DAY_OF_MONTH_13	0.00

Figure 64:Case 2 result using XGBoost

Removing the Feature(s) that Bias the accuracy of model prediction (only for AA-DFW)

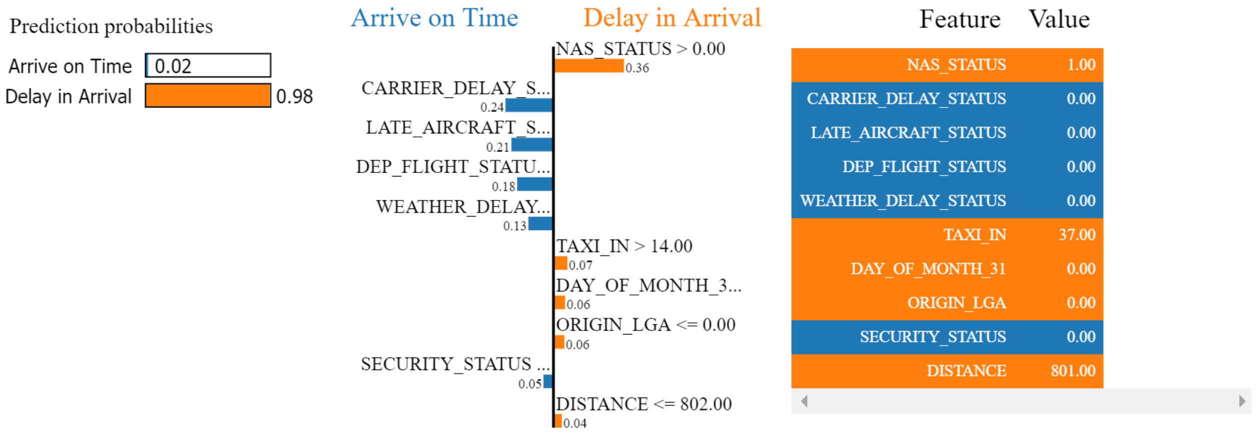
Here Random Forest model was run with a data set where Flight Number was included as a feature and LIME provided the following prediction and feature importance.



Feature	Value
NAS_STATUS	1.00
OP_CARRIER_FL_NUM_2290	0.00
OP_CARRIER_FL_NUM_1399	0.00
OP_CARRIER_FL_NUM_2830	0.00
OP_CARRIER_FL_NUM_877	0.00
OP_CARRIER_FL_NUM_972	0.00
OP_CARRIER_FL_NUM_1224	0.00
OP_CARRIER_FL_NUM_1209	0.00
OP_CARRIER_FL_NUM_2404	0.00
OP_CARRIER_FL_NUM_882	0.00

Figure 65: Random Forest model was run with Flight Number

Model predicted that this flight will be delayed but the top feature that the model considered are all different flight numbers (except NAS delay) which is not very intuitive. Same model was run after dropping Flight Number Feature and LIME provided the following interpretation and the prediction probability increased by 5%.



Feature	Value
NAS_STATUS	1.00
CARRIER_DELAY_STATUS	0.00
LATE_AIRCRAFT_STATUS	0.00
DEP_FLIGHT_STATUS	0.00
WEATHER_DELAY_STATUS	0.00
TAXI_IN	37.00
DAY_OF_MONTH_31	0.00
ORIGIN_LGA	0.00
SECURITY_STATUS	0.00
DISTANCE	801.00

Figure 66: Random Forest model was run without Flight Number

Airline customer centric interpretable prediction (Only for AA-DFW)

As a customer some of the features used as an input may not be intuitive. For an example, taxi out time is a feature that a customer may not think about before entering the airport. So, if a customer wants to know whether their flight will be delayed or not and what the probable cause of that delay is by inputting some features that they already know can help them plan better. Here a random forest model ran with the following features.

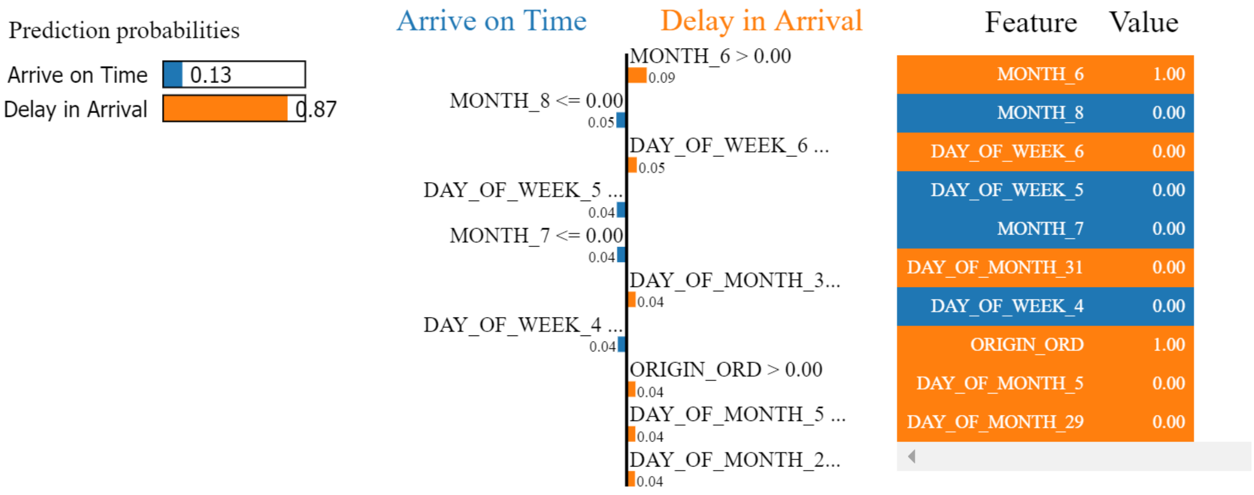
- Distance
- Origin
- Day of the Week
- Day of the Month
- Month
- Quarter of the year

These features are easy for the customer to input in the model to get the prediction. With this small amount of features, the prediction probability suffers a lot, however a customer can still check the probability and probable cause of the delay.

Case-1: Delay flight

Inputs:

- Distance: 801.0
- Origin: ORD
- Day of the Week: Saturday (6)
- Day of the Month: 12
- Month: June (6)
- Quarter of the year: 2



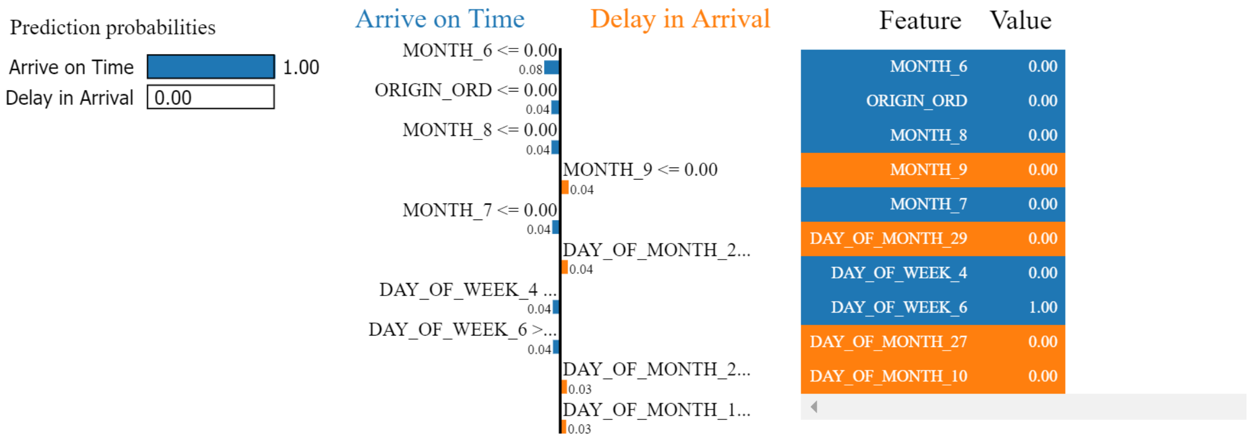
Feature	Value
MONTH_6	1.00
MONTH_8	0.00
DAY_OF_WEEK_6	0.00
DAY_OF_WEEK_5	0.00
MONTH_7	0.00
DAY_OF_MONTH_31	0.00
DAY_OF_WEEK_4	0.00
ORIGIN_ORD	1.00
DAY_OF_MONTH_5	0.00
DAY_OF_MONTH_29	0.00

Figure 67: Case 1: Delay flight based on customer centric Interpretable prediction

Case-2: Flight Arrive on time

Inputs:

- Distance: 936.0
- Origin: CLT
- Day of the Week: Saturday (6)
- Day of the Month: 30
- Month: January (1)
- Quarter of the year: 1



Feature	Value
MONTH_6	0.00
ORIGIN_ORD	0.00
MONTH_8	0.00
MONTH_9	0.00
MONTH_7	0.00
DAY_OF_MONTH_29	0.00
DAY_OF_WEEK_4	0.00
DAY_OF_WEEK_6	1.00
DAY_OF_MONTH_27	0.00
DAY_OF_MONTH_10	0.00

Figure 68 Case 2: Flight arrive on time based on customer centric Interpretable prediction

For Case-1, features that influence the prediction include the month of travel of that flight and the origin (ORD). During our exploratory data analysis, we observed percentage of flight delays are higher for summer and ORD Airport.

For Case-2, model predicted with very high confidence that flight will arrive on time as it is not scheduled on the month of June and is not departing from ORD.

Chapter 5

CONCLUSION

During this master’s thesis, different prediction models and various evaluation methods are explored using different applications and services. By using historic data, interesting results were observed on the predictability of delays. The best delay prediction method emerged to be the most specific one, which takes into account all the combinations of categorical features.

The performances of the models were interesting to evaluate, due to the numerous features used. From the exploratory data analysis (EDA), we found that AA flights departing from ORD and arriving at DFW are the most delayed flights from the exploratory data analysis (EDA), we found that AA flights departing from ORD and arriving at DFW are the most delayed flights. For southwest airlines departing from LAX had the most total number of delays. There is a relationship between arrival and departure delay. Also, found June and July are the worse month when it comes to total number of delays.

For AA, model accuracy using Python for Random Forest obtained 85% accuracy, Adaboost obtained 83% accuracy, XGBoost obtained 90% accuracy, Gradient Boost obtained 88% accuracy and MLP obtained 86% accuracy. For WN, model accuracy using Python for Random Forest obtained 81% accuracy, Adaboost obtained 83% accuracy, XGBoost obtained 86% accuracy, Gradient Boost 84% accuracy and MLP obtained 86% accuracy. The accuracy for AA of the random forest model using Azure Studio for is 96%, and Azure- AutoML gives 82% for XGBoost and 92% for Random Forest. The differences in accuracy between Python and Azure AutoML may be the result of the data pre-processing, as Azure AutoML automates the machine learning workflow (that includes data preprocess) on the other hand for Python model data preprocess requires domain knowledge.

In the business world, we can save time and money by improving the understanding of our machine learning model prediction. Using LIME, it is possible to find out the cause of delay from the prediction model and take action to mitigate those reasons.

REFERENCE

Airlines For America | Annual U.S. Impact of Flight Delays (NEXTOR report). (n.d.). Retrieved January 3, 2021, from https://www.airlines.org/data/annual-u-s-impact-of-flight-delays-nextor-report/

Apo-. (2016). *Cost of Delay Estimates FAA APO-100*.

Baik, H., Li, T., & Chintapudi, N. K. (2010). Estimation of Flight Delay Costs for U.S. Domestic Air Passengers. *Transportation Research Record: Journal of the Transportation Research Board*, *2177*(1), 49–59. https://doi.org/10.3141/2177-07

Bank Data: SMOTE. This will be a short post before we… | by Zaki Jefferson | Analytics Vidhya | Medium. (n.d.). Retrieved April 22, 2021, from https://medium.com/analytics-vidhya/bank-data-smote-b5cb01a5e0a2

Bureau of Transportation Statistics. (n.d.). Retrieved January 3, 2021, from https://www.bts.gov/

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

Data Mining for Business Analytics: Concepts, Techniques, and Applications in R | Wiley. (n.d.). Retrieved February 23, 2021, from https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+in+R-p-9781118879368

Gui, G., Liu, F., Sun, J., Yang, J., Zhou, Z., & Zhao, D. (2020). Flight delay prediction based on aviation big data and machine learning. *IEEE Transactions on Vehicular Technology*, *69*(1), 140–150. https://doi.org/10.1109/TVT.2019.2954094

Importing Data in Python. (n.d.). Retrieved February 28, 2021, from https://www.tutorialspoint.com/importing-data-in-python

Interpretable Machine Learning - Christoph Molnar - Google Books. (n.d.). Retrieved April 22, 2021, from https://books.google.com/books?hl=en&lr=&id=jBm3DwAAQBAJ&oi=fnd&pg=PP1&dq=Molnar,+C.+(2020).+Interpretable+machine+learning.+Lulu.+Com&ots=EgsYVrECS5&sig=VlidO835EGnXahwu3gIJ-R2-ONY#v=onepage&q=Molnar%2C C. (2020). Interpretable machine learning. Lulu. Com&f=false

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2000). An introduction to Statistical Learning. In *Current medicinal chemistry* (Vol. 7, Issue 10). https://doi.org/10.1007/978-1-4614-7138-7

Kalliguddi, A., Leboulluec, A., Kalliguddi, A. M., & Leboulluec, A. K. (2017). Predictive Modeling of Aircraft Flight Delay Adaptive Interdisciplinary Pain Management View project Predictive Modeling of Aircraft Flight Delay. *Article in Universal Journal of Management*, *5*(10), 485–491. https://doi.org/10.13189/ujm.2017.051003

Kuhn, N., & Jamadagni, N. (2017). Application of Machine Learning Algorithms to Predict Flight Arrival Delays. *Cs229*, 1–6.

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

Masís, S. (2021). *Interpretable Machine Learning with Python: Learn to build interpretable high-performance models with hands-on real-world*. Packt Publishing.

Rebollo, J. J., & Balakrishnan, H. (2014). Characterization and prediction of air traffic delays. *Transportation Research Part C: Emerging Technologies*, *44*, 231–241. https://doi.org/10.1016/j.trc.2014.04.007

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). “Why should i trust you?” Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 1135–1144. https://doi.org/10.1145/2939672.2939778

Shmueli, G., Bruce, P., Yahav, I., Patel, N., & Lichtendahl Jr., K. (2017). *tutorial*. John Wiley & Sons.

Teja, S. (n.d.). *(PDF) Flight Delay Prediction Using Machine Learning Algorithm XGBoost*. Retrieved April 2, 2021, from https://www.researchgate.net/publication/344227817_Flight_Delay_Prediction_Using_Machine_Learning_Algorithm_XGBoost

Truong, D., Friend, M. A., & Chen, H. (2018). Applications of business analytics in predicting flight on-time performance in a complex and dynamic system. *Transportation Journal*, *57*(1), 24–52. https://doi.org/10.5325/transportationj.57.1.0024

Yazdi, M. F., Kamel, S. R., Chabok, S. J. M., & Kheirabadi, M. (2020a). Flight delay prediction based on deep learning and Levenberg-Marquart algorithm. *Journal of Big Data*, *7*(1), 1–28. https://doi.org/10.1186/s40537-020-00380-z

Yazdi, M. F., Kamel, S. R., Chabok, S. J. M., & Kheirabadi, M. (2020b). Flight delay prediction based on deep learning and Levenberg-Marquart algorithm. *Journal of Big Data*, *7*(1), 106. https://doi.org/10.1186/s40537-020-00380-z

APPENDIX

Gathering Data

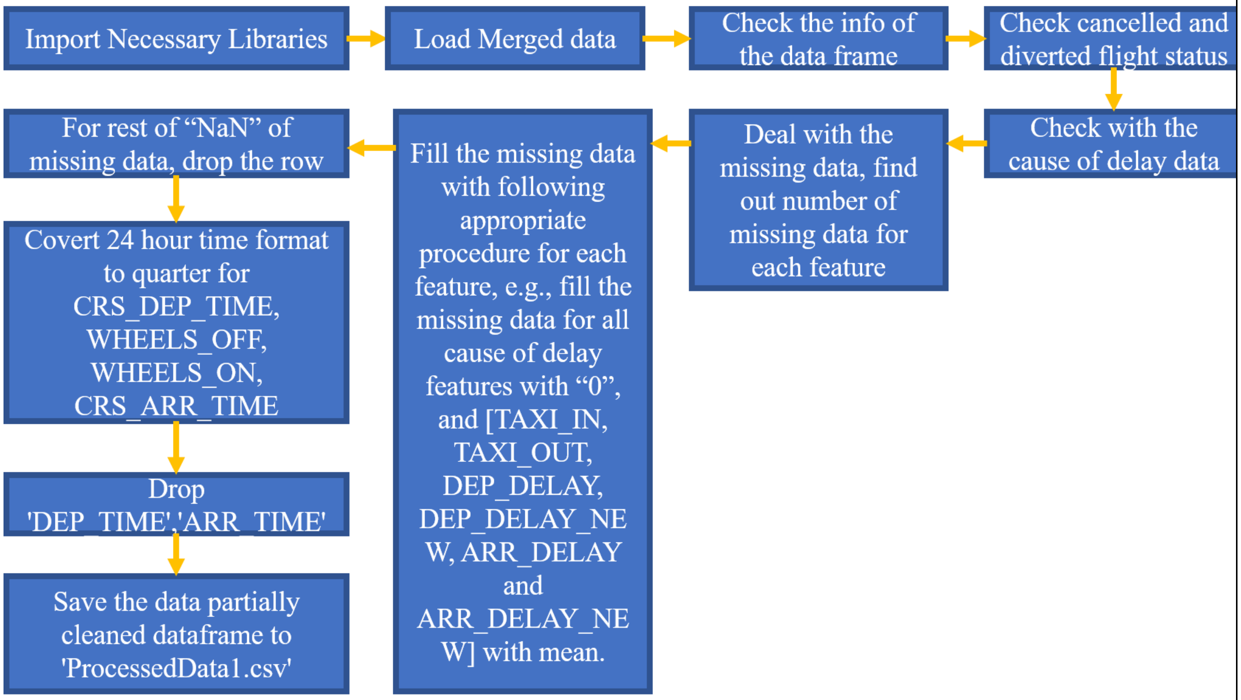
```
# # Gathering data
# ### Data were downloaded from BTS website. 2016-2019. 48 months

#import libraries
import pandas as pd
#Glob is a general term used to define techniques to
#match specified patterns according to rules related to Unix shell
from glob import glob
import datetime, warnings, scipy
warnings.filterwarnings("ignore")
#make list of flies in the folder
#careful about file path it is "/"
stock_files = sorted (glob('C:/Users/raihanm/Desktop/FD/data_1/*.csv'))
stock_files
df_from_each_file = (pd.read_csv(f, sep=',') for f in stock_files)
df_merged   = pd.concat(df_from_each_file, ignore_index=True)
df_merged.to_csv( "merged.csv")
```

```
df1=df_merged
df1.columns

df1=df1.drop(['FIRST_DEP_TIME','TOTAL_ADD_GTIME','Unnamed: 37'],axis=1)
df1.shape
filt=(df1['DEST']=='DFW')
df2=df1[filt]
filt2=(df2['OP_UNIQUE_CARRIER']=='AA')
df3=df2[filt2]
origin_list=['JFK','LGA','ATL','CLT','PHL','DCA','ORD','LAX','MIA','PHX','DEN']
df4=df3[df3['ORIGIN'].isin(origin_list)]
df4.head()
df4.shape
df4.to_csv('dfm.csv')
```

Data Cleaning: Part 1

Data Cleaning: Part 1

Below are the steps that are taken for data cleaning:



Import Necessary Libraries

Load Merged data

Check the info of the data frame

Check cancelled and diverted flight status

Check with the cause of delay data

Deal with the missing data, find out number of missing data for each feature

Fill the missing data with following appropriate procedure for each feature, e.g., fill the missing data for all cause of delay features with “0”, and [TAXI_IN, TAXI_OUT, DEP_DELAY, DEP_DELAY_NEW, ARR_DELAY and ARR_DELAY_NEW] with mean.

For rest of “NaN” of missing data, drop the row

Covert 24 hour time format to quarter for CRS_DEP_TIME, WHEELS_OFF, WHEELS_ON, CRS_ARR_TIME

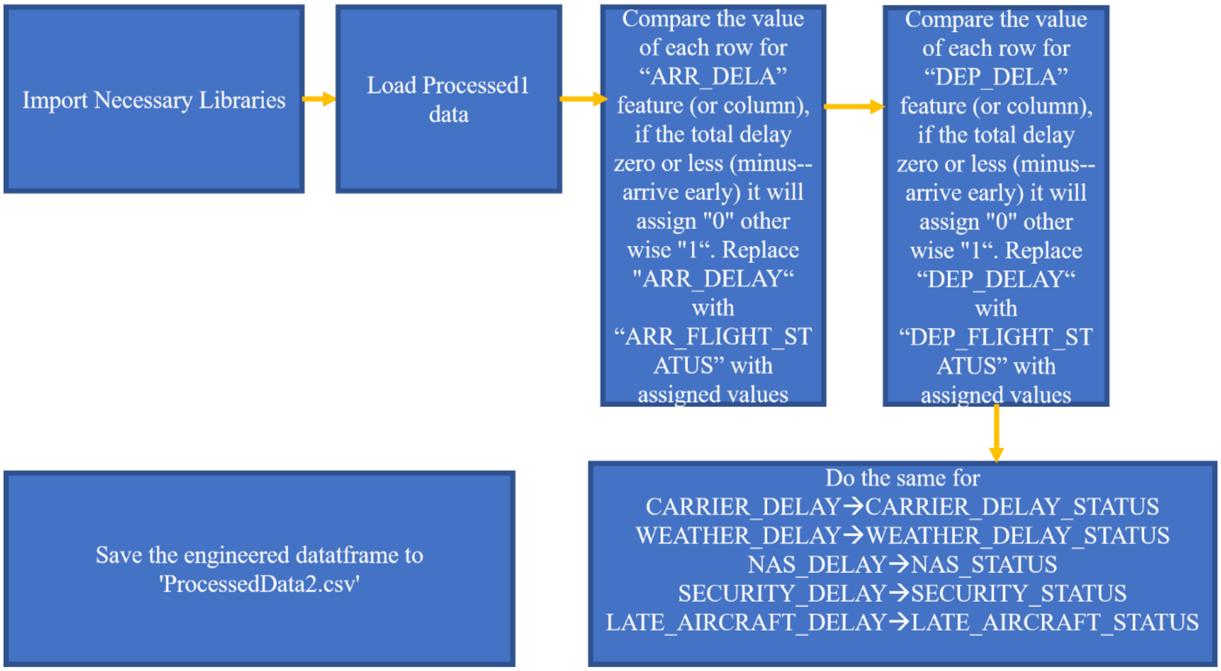
Drop 'DEP_TIME','ARR_TIME'

Save the data partially cleaned dataframe to 'ProcessedData1.csv'

Feature Engineering

Data Engineering

Below are the steps that are taken for data engineering:



Import Necessary Libraries

Load Processed1 data

Compare the value of each row for “ARR_DELA” feature (or column), if the total delay zero or less (minus-- arrive early) it will assign "0" other wise "1“. Replace "ARR_DELAY“ with “ARR_FLIGHT_STATUS” with assigned values

Compare the value of each row for “DEP_DELA” feature (or column), if the total delay zero or less (minus-- arrive early) it will assign "0" other wise "1“. Replace “DEP_DELAY“ with “DEP_FLIGHT_STATUS” with assigned values

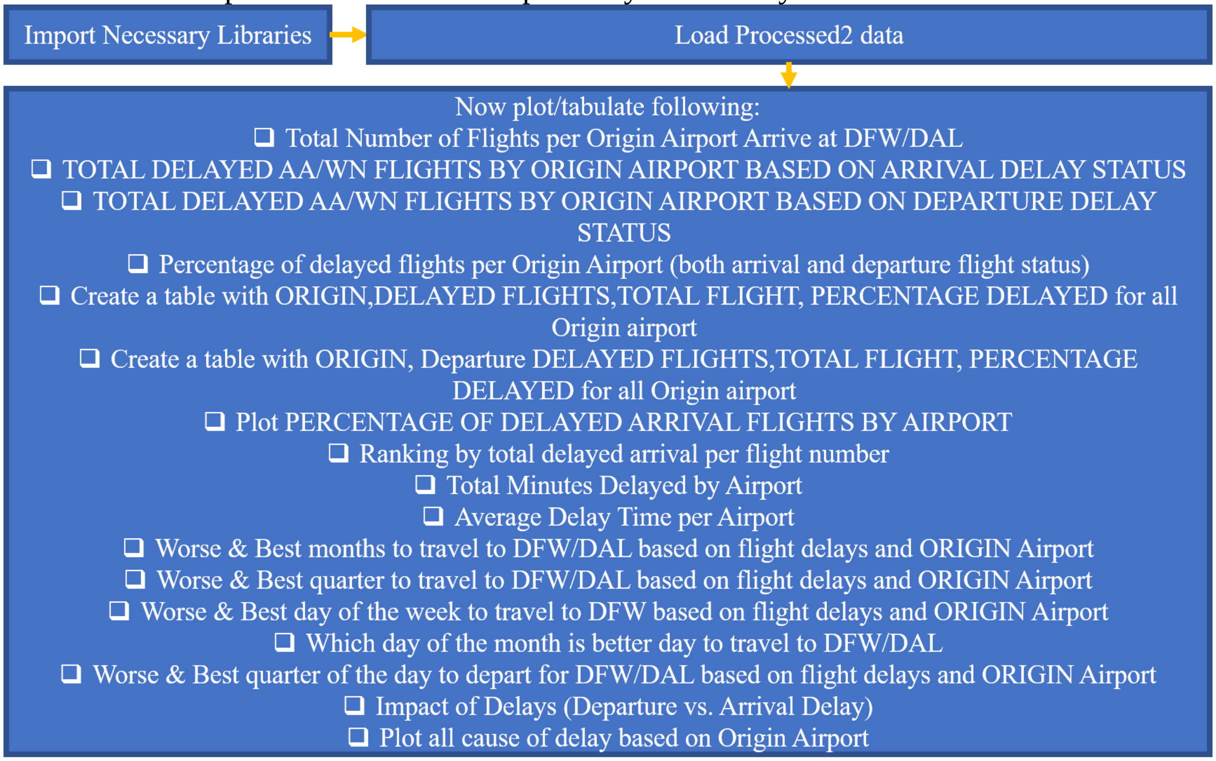
Do the same for
CARRIER_DELAY→CARRIER_DELAY_STATUS
WEATHER_DELAY→WEATHER_DELAY_STATUS
NAS_DELAY→NAS_STATUS
SECURITY_DELAY→SECURITY_STATUS
LATE_AIRCRAFT_DELAY→LATE_AIRCRAFT_STATUS

Save the engineered datatframe to 'ProcessedData2.csv'

Exploratory Data Analysis

Exploratory Data Analysis

Below are the steps that are taken for Exploratory Data Analysis :



Import Necessary Libraries

Load Processed2 data

Now plot/tabulate following:

□ Total Number of Flights per Origin Airport Arrive at DFW/DAL

□ TOTAL DELAYED AA/WN FLIGHTS BY ORIGIN AIRPORT BASED ON ARRIVAL DELAY STATUS

□ TOTAL DELAYED AA/WN FLIGHTS BY ORIGIN AIRPORT BASED ON DEPARTURE DELAY STATUS

□ Percentage of delayed flights per Origin Airport (both arrival and departure flight status)

□ Create a table with ORIGIN,DELAYED FLIGHTS,TOTAL FLIGHT, PERCENTAGE DELAYED for all Origin airport

□ Create a table with ORIGIN, Departure DELAYED FLIGHTS,TOTAL FLIGHT, PERCENTAGE DELAYED for all Origin airport

□ Plot PERCENTAGE OF DELAYED ARRIVAL FLIGHTS BY AIRPORT

□ Ranking by total delayed arrival per flight number

□ Total Minutes Delayed by Airport

□ Average Delay Time per Airport

□ Worse & Best months to travel to DFW/DAL based on flight delays and ORIGIN Airport

□ Worse & Best quarter to travel to DFW/DAL based on flight delays and ORIGIN Airport

□ Worse & Best day of the week to travel to DFW based on flight delays and ORIGIN Airport

□ Which day of the month is better day to travel to DFW/DAL

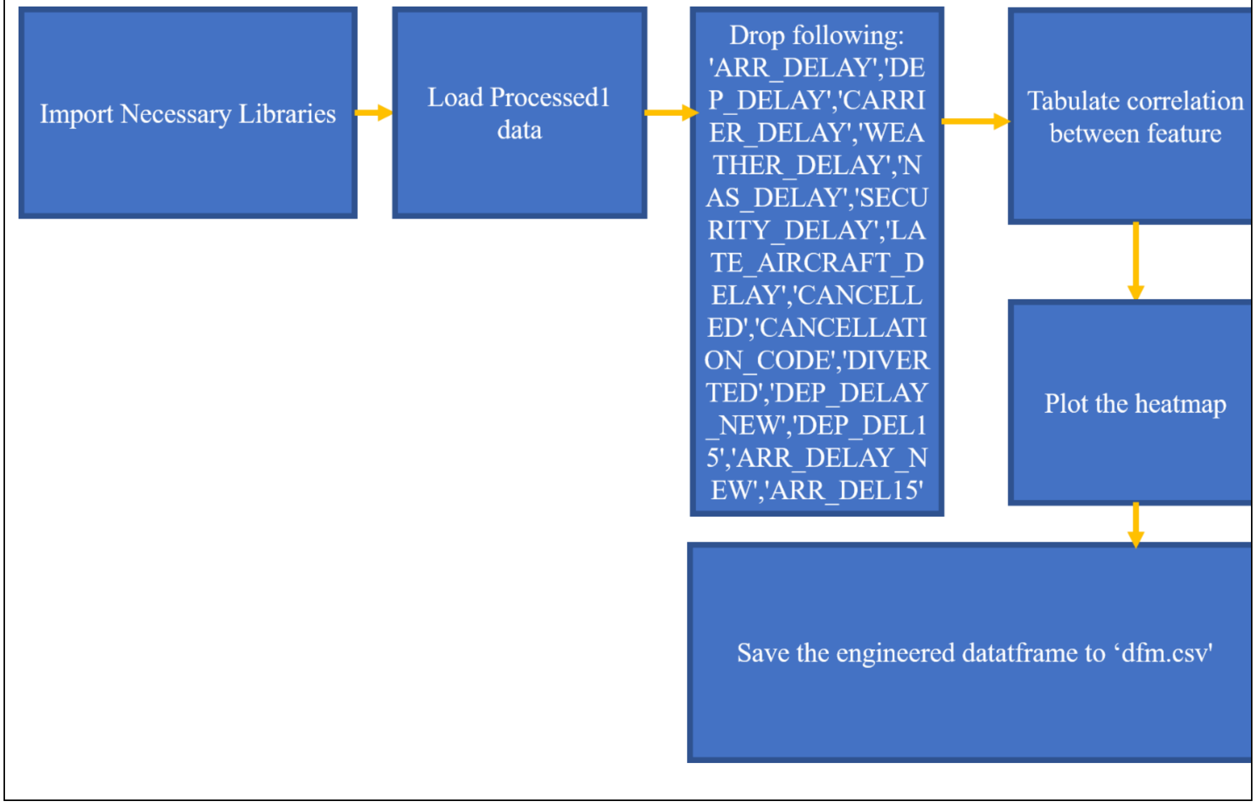
□ Worse & Best quarter of the day to depart for DFW/DAL based on flight delays and ORIGIN Airport

□ Impact of Delays (Departure vs. Arrival Delay)

□ Plot all cause of delay based on Origin Airport

Data cleaning: Part 2

Below are the steps that are taken for Data cleaning- Part 2:


Import Necessary Libraries
Load Processed1 data
Drop following: 'ARR_DELAY','DEP_DELAY','CARRIER_DELAY','WEATHER_DELAY','NAS_DELAY','SECURITY_DELAY','LATE_AIRCRAFT_DELAY','CANCELLED','CANCELLATION_CODE','DIVERTED','DEP_DELAY_NEW','DEP_DEL15','ARR_DELAY_NEW','ARR_DEL15'
Tabulate correlation between feature
Plot the heatmap
Save the engineered datatframe to ‘dfm.csv'

Final Data Processing for Modeling

```
# # Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns
sns.set_style('darkgrid')
pd.set_option('display.max_columns', None)
import datetime, warnings, scipy
warnings.filterwarnings("ignore")

from sklearn import metrics, linear_model
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
plot_confusion_matrix
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict,
RepeatedStratifiedKFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
from scipy.optimize import curve_fit
from sklearn.svm import SVC
from random import sample

import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import StandardScaler, LabelBinarizer

import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
import scipy.stats as stats


# # Data Loading & QC

dfm = pd.read_csv('dfm.csv', index_col=0)
dfm.head()
dfm.info()
dfm.YEAR.value_counts(normalize=True)

dfm.QUARTER.value_counts(normalize=True)





dfm.MONTH.value_counts(normalize=True)


dfm.DAY_OF_MONTH.value_counts(normalize=True)
dfm.DAY_OF_WEEK.value_counts(normalize=True)
dfm.OP_CARRIER_FL_NUM.nunique()
dfm.OP_CARRIER_FL_NUM.value_counts(normalize=True)
dfm.head(2)
dfm.ORIGIN.value_counts(normalize=True)


dfm.CRS_DEP_TIME.value_counts(normalize=True)

dfm.TAXI_OUT.nunique()

dfm.TAXI_OUT.value_counts(normalize=True)

dfm.head(2)

dfm.WHEELS_OFF.value_counts(normalize=True)

dfm.WHEELS_ON     .value_counts(normalize=True)

dfm.TAXI_IN.nunique()

dfm.TAXI_IN.value_counts(normalize=True)

dfm.CRS_ARR_TIME.value_counts(normalize=True)

dfm.head(2)

dfm.CRS_ELAPSED_TIME.nunique()

dfm.CRS_ELAPSED_TIME.value_counts(normalize=True)


dfm.ACTUAL_ELAPSED_TIME.nunique()
dfm.ACTUAL_ELAPSED_TIME.value_counts(normalize=True)

dfm.head(2)

dfm.AIR_TIME.nunique()

dfm.AIR_TIME.value_counts(normalize=True)

dfm.DISTANCE.nunique()

dfm.DISTANCE.value_counts(normalize=True)

dfm.ARR_FLIGHT_STATUS.nunique()




dfm.ARR_FLIGHT_STATUS.value_counts(normalize=True)

dfm.head(2)

dfm.CARRIER_DELAY_STATUS.value_counts(normalize=True)

dfm.WEATHER_DELAY_STATUS.value_counts(normalize=True)

dfm.NAS_STATUS.value_counts(normalize=True)

dfm.SECURITY_STATUS.value_counts(normalize=True)

dfm.LATE_AIRCRAFT_STATUS.value_counts(normalize=True)


# # Data Distribution
#
# Based on the percentage of delayed flight we know that this dataset is unbalanced, but just to follow standard workflows we will write a short function to see this visually and then keep on going.
#

def scaling_check(data):

    case_count = dfm['ARR_FLIGHT_STATUS'].value_counts() # 'data' is our input which will be any of the 3 dataframes created
    print('Legend:')
    print(case_count)
```

```
   plt.figure(figsize=(10,6))
   sns.barplot(x=case_count.index, y=case_count.values)
   plt.rcParams["figure.facecolor"] = "blue"
   plt.title('Data Distribution', fontsize=16)
   plt.xlabel('Arrival Flight Status', fontsize=12)
   plt.ylabel('Number of Flights', fontsize=12)
   plt.xticks(range(len(case_count.index)), ['ON TIME(0)', 'DELAYED(1)'])
   plt.show()


scaling_check (dfm)

dfm.ARR_FLIGHT_STATUS.value_counts(normalize=True)


# ## Calculating the weight to use later for the imbalanced dataset

df_test=dfm['ARR_FLIGHT_STATUS'].value_counts()


df_test.head()


count_0=df_test[0]

count_0

count_1=df_test[1]


count_1

initial_bias = np.log([count_1/count_0])
initial_bias

weight_for_0 = (1/count_0)*(count_0 + count_1)/2.0
weight_for_1 = (1/count_1)*(count_0 + count_1)/2.0

class_weight = {0: weight_for_0, 1: weight_for_1}

print('Weight for class 0: {:.2f}'.format(weight_for_0))
print('Weight for class 1: {:.2f}'.format(weight_for_1))
```

```
# Before going into the modeling, we will create histograms for all the features to get a better
feeling of them:

dfm.hist(figsize  = [15, 15],bins=9)
plt.show()


dfm.columns



fig, axes = plt.subplots(nrows=5, ncols=3, figsize=(16,14), sharey=True)

categoricals = ['QUARTER', 'MONTH', 'DAY_OF_MONTH', 'DAY_OF_WEEK',
              'OP_CARRIER_FL_NUM', 'ORIGIN', 'WHEELS_OFF', 'WHEELS_ON',
              'DEP_FLIGHT_STATUS','CARRIER_DELAY_STATUS',
'WEATHER_DELAY_STATUS',
              'NAS_STATUS','SECURITY_STATUS', 'LATE_AIRCRAFT_STATUS']

for col, ax in zip(categoricals, axes.flatten()):
    (dfm.groupby(col).sum()['ARR_FLIGHT_STATUS'].sort_values().plot.bar(ax=ax))

    ax.set_title(col)

fig.tight_layout()

dfm.head()





QUARTER_dummies = pd.get_dummies(dfm['QUARTER'], prefix='QUARTER',
drop_first=True)
MONTH_dummies = pd.get_dummies(dfm['MONTH'], prefix='MONTH', drop_first=True)
DAY_OF_MONTH_dummies = pd.get_dummies(dfm['DAY_OF_MONTH'],
prefix='DAY_OF_MONTH', drop_first=True)
DAY_OF_WEEK_dummines = pd.get_dummies(dfm['DAY_OF_WEEK'],
prefix='DAY_OF_WEEK', drop_first=True)
#OP_CARRIER_FL_NUM_dummies = pd.get_dummies(dfm['OP_CARRIER_FL_NUM'],
prefix='OP_CARRIER_FL_NUM', drop_first=True)
ORIGIN_dummies = pd.get_dummies(dfm['ORIGIN'], prefix='ORIGIN', drop_first=True)
CRS_DEP_TIME_dummies = pd.get_dummies(dfm['CRS_DEP_TIME'],
prefix='CRS_DEP_TIME', drop_first=True)
WHEELS_OFF_dummies = pd.get_dummies(dfm['WHEELS_OFF'],
prefix='WHEELS_OFF', drop_first=True)
```

```
WHEELS_ON_dummies = pd.get_dummies(dfm['WHEELS_ON'], prefix='WHEELS_ON',
drop_first=True)
CRS_ARR_TIME_dummies = pd.get_dummies(dfm['CRS_ARR_TIME'],
prefix='CRS_ARR_TIME', drop_first=True)


dfm = dfm.drop(['YEAR','OP_UNIQUE_CARRIER','DEST','QUARTER', 'MONTH',
'DAY_OF_MONTH', 'DAY_OF_WEEK', 'OP_CARRIER_FL_NUM', 'ORIGIN',
'CRS_DEP_TIME',
            'WHEELS_OFF', 'WHEELS_ON',
'CRS_ARR_TIME','DEP_FLIGHT_STATUS'],axis=1)

dfm =
pd.concat([dfm,QUARTER_dummies,MONTH_dummies,DAY_OF_MONTH_dummies,DA
Y_OF_WEEK_dummines,
             ORIGIN_dummies,CRS_DEP_TIME_dummies,

WHEELS_OFF_dummies,WHEELS_ON_dummies,CRS_ARR_TIME_dummies],axis=1)


dfm.head()


dfm.shape

dfm.to_csv('dfm_ready3.csv')
```

Checking data for imbalance

```
# # Performance metrics with and without class weight

# # Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns
sns.set_style('darkgrid')
pd.set_option('display.max_columns', None)
import datetime, warnings, scipy
warnings.filterwarnings("ignore")

from sklearn import metrics, linear_model
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
plot_confusion_matrix
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict,
RepeatedStratifiedKFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
from scipy.optimize import curve_fit
from sklearn.svm import SVC
from random import sample

import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler, LabelBinarizer

import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
import scipy.stats as stats

# # Data Loading & QC

dfm = pd.read_csv('dfm_ready3', index_col=0)
dfm.head()
dfm.ARR_FLIGHT_STATUS.nunique()

dfm.ARR_FLIGHT_STATUS.value_counts(normalize=True)


# Create features (X) and labels (y)
y = dfm['ARR_FLIGHT_STATUS']
X = dfm.drop(['ARR_FLIGHT_STATUS'], axis=1)

# Perform the split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# ## Random Forest With No Class Weighting
#

# I'll start by instantiating the RandomForestClassifier
forest = RandomForestClassifier(n_estimators=100, max_depth=5)
forest.fit(X_train, y_train)

# Now to check the accuracy score
forest.score(X_train, y_train)

# And for the test set:
forest.score(X_test, y_test)

# ### MODEL PERFORMANCE

# Test set predictions
pred_rf = forest.predict(X_test)

# confusion matrix and classfication report
print('\nConfusion Matrix')
print('----------------')
pd.crosstab(y_test, pred_rf, rownames=['True'], colnames=['Predicted'], margins=True)

print('Classificiation Report')
print('---------------------')
print(classification_report(y_test, pred_rf))

print("Testing Accuracy for RandomForest Classifier: {:.4}%".format(accuracy_score(y_test,
pred_rf) * 100))

# ### Random Forest With Class Weighting
forest_cw = RandomForestClassifier(n_estimators=100, max_depth=5,
class_weight='balanced')
forest_cw.fit(X_train, y_train)
```

```
# Now to check the accuracy score
forest_cw.score(X_train, y_train)

# And for the test set:
forest_cw.score(X_test, y_test)


# ### MODEL PERFORMANCE

# Test set predictions
pred_rfcw = forest_cw.predict(X_test)

# confusion matrix and classfication report
print('\nConfusion Matrix')
print('----------------')
pd.crosstab(y_test, pred_rfcw, rownames=['True'], colnames=['Predicted'], margins=True)

print('Classificiation Report')
print('---------------------')
print(classification_report(y_test, pred_rfcw))

print("Testing Accuracy for RandomForest Classifier: {:.4}%".format(accuracy_score(y_test,
pred_rfcw) * 100))

# ### Random Forest With Bootstrat Class Weighting

forest_bcw = RandomForestClassifier(n_estimators=100, max_depth=5,
class_weight='balanced_subsample')
forest_bcw.fit(X_train, y_train)

# Now to check the accuracy score
forest_bcw.score(X_train, y_train)

# And for the test set:
forest_bcw.score(X_test, y_test)

# ### MODEL PERFORMANCE

# Test set predictions
pred_rfbcw = forest_bcw.predict(X_test)

# confusion matrix and classfication report
print('\nConfusion Matrix')
print('----------------')
pd.crosstab(y_test, pred_rfbcw, rownames=['True'], colnames=['Predicted'], margins=True)
```

```
print('Classificiation Report')
print('---------------------')
print(classification_report(y_test, pred_rfbcw))

print("Testing Accuracy for RandomForest Classifier: {:.4}%".format(accuracy_score(y_test,
pred_rfbcw) * 100))
```

Model and Interpretation

```
# Importing Required Libraries
import numpy as np
import pandas as pd
import os
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
import xgboost as xgb
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix, classification_report
pd.set_option('display.max_columns',None)

data = pd.read_csv("dfm_ready_cv.csv")
data.head()

data.columns

data=data.drop(['Unnamed: 0'],axis=1)

data.head()

train,test=train_test_split(data,test_size=0.3,random_state=0,stratify=data['ARR_FLIGHT_STATUS'])

train.head()

test.head()

train.shape

test.shape

# Create X_train,Y_train,X_test, Y_test
X_train = train.drop(['ARR_FLIGHT_STATUS'], axis=1)
Y_train = train['ARR_FLIGHT_STATUS']

X_test  = test.drop(['ARR_FLIGHT_STATUS'], axis=1)
Y_test  = test['ARR_FLIGHT_STATUS']
```

```
# Random Forest
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)
random_forest_preds = random_forest.predict(X_test)
print('The accuracy of the Random Forests model is
:\t',metrics.accuracy_score(random_forest_preds,Y_test))


import lime
import lime.lime_tabular

predict_fn_rf = lambda x: random_forest.predict_proba(x).astype(float)
X = X_train.values
explainer = lime.lime_tabular.LimeTabularExplainer(X,feature_names =
X_train.columns,class_names=['Arrive on Time','Delay in Arrival'],kernel_width=5)

test.loc[[112707]]

choosen_instance = X_test.loc[[112707]].values[0]
exp = explainer.explain_instance(choosen_instance, predict_fn_rf,num_features=10)
exp.show_in_notebook(show_all=False)

test.loc[[698]]

choosen_instance = X_test.loc[[698]].values[0]
exp = explainer.explain_instance(choosen_instance, predict_fn_rf,num_features=10)
exp.show_in_notebook(show_all=False)

adaboost_clf = AdaBoostClassifier(random_state=42)
adaboost_clf.fit(X_train, Y_train)
adaboost_clf_preds = adaboost_clf.predict(X_test)
print('The accuracy of the Adaboost model is
:\t',metrics.accuracy_score(adaboost_clf_preds,Y_test))


predict_fn_ab = lambda x: adaboost_clf.predict_proba(x).astype(float)
X = X_train.values
explainer = lime.lime_tabular.LimeTabularExplainer(X,feature_names =
X_train.columns,class_names=['Arrive on Time','Delay in Arrival'],kernel_width=5)


choosen_instance = X_test.loc[[112707]].values[0]
exp = explainer.explain_instance(choosen_instance, predict_fn_ab,num_features=10)
exp.show_in_notebook(show_all=False)
```

```
choosen_instance = X_test.loc[[698]].values[0]
exp = explainer.explain_instance(choosen_instance, predict_fn_ab,num_features=10)
exp.show_in_notebook(show_all=False)

gbt_clf = GradientBoostingClassifier(random_state=42)
gbt_clf.fit(X_train, Y_train)
gbt_clf_preds = gbt_clf.predict(X_test)
print('The accuracy of the Gradient boost model is
:\t',metrics.accuracy_score(gbt_clf_preds,Y_test))

predict_fn_gbt = lambda x: gbt_clf.predict_proba(x).astype(float)
X = X_train.values
explainer = lime.lime_tabular.LimeTabularExplainer(X,feature_names =
X_train.columns,class_names=['Arrive on Time','Delay in Arrival'],kernel_width=5)

choosen_instance = X_test.loc[[112707]].values[0]
exp = explainer.explain_instance(choosen_instance, predict_fn_gbt,num_features=10)
exp.show_in_notebook(show_all=False)

choosen_instance = X_test.loc[[698]].values[0]
exp = explainer.explain_instance(choosen_instance, predict_fn_gbt,num_features=10)
exp.show_in_notebook(show_all=False)

xgb_clf = xgb.XGBClassifier()
xgb_clf.fit(X_train, Y_train)
xgb_clf_preds = clf.predict(X_test)
print('The accuracy of the XGboost model is
:\t',metrics.accuracy_score(xgb_clf_preds,Y_test))

predict_fn_xgb = lambda x: xgb_clf.predict_proba(x).astype(float)
X = X_train.values
explainer = lime.lime_tabular.LimeTabularExplainer(X,feature_names =
X_train.columns,class_names=['Arrive on Time','Delay in Arrival'],kernel_width=5)

choosen_instance = X_test.loc[[112707]].values[0]
exp = explainer.explain_instance(choosen_instance, predict_fn_xgb,num_features=10)
exp.show_in_notebook(show_all=False)

choosen_instance = X_test.loc[[698]].values[0]
exp = explainer.explain_instance(choosen_instance, predict_fn_xgb,num_features=10)
exp.show_in_notebook(show_all=False)
```

MLP-NN

```
import pandas as pd
import numpy as np
np.random.seed(0)
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns
import xgboost as xgb
sns.set_style('darkgrid')
pd.set_option('display.max_columns', None)
import datetime, warnings, scipy
warnings.filterwarnings("ignore")

import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler, LabelBinarizer
from sklearn.model_selection import train_test_split
from keras.layers import Dense, Conv2D , SeparableConv2D, MaxPool2D, Flatten , Dropout ,
BatchNormalization
from keras import Sequential
from keras.layers import Dense
from sklearn import preprocessing
import sklearn.metrics as metrics
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score

import tensorflow as tf
from tensorflow import keras
dfm_ready = pd.read_csv('dfm_ready3.csv', index_col=0)
dfm_ready.head().append(dfm_ready.tail())


# The next cell will make sure that all my features are in floating format, followed by a double
check with the .info() function

df = dfm_ready.astype(float)


df.info()


# # Function Library
```

```
def visualize_training_results(results):
    history = results.history
    plt.figure()
    plt.plot(history['val_loss'])
    plt.plot(history['loss'])
    plt.legend(['val_loss', 'loss'])
    plt.title('LOSS', fontsize=14)
    plt.xlabel('Epochs', fontsize=12)
    plt.ylabel('Loss', fontsize=12)
    plt.show()

    plt.figure()
    plt.plot(history['val_accuracy'])
    plt.plot(history['accuracy'])
    plt.legend(['val_accuracy', 'accuracy'])
    plt.title('ACCURACY', fontsize=14)
    plt.xlabel('Epochs', fontsize=12)
    plt.ylabel('Accuracy', fontsize=12)
    plt.show()


def conf_matrix(conf_mat,):

    conf_mat = pd.DataFrame(conf_mat, index = ['0', '1'], columns = ['0', '1'])
    sns.set(font_scale=1.4)
    plt.figure(figsize = (8,7))
    sns.heatmap(conf_mat, cmap= "Blues", linecolor = 'black' , linewidth = 1, annot = True,
fmt=")
    plt.xlabel('Predicted', fontsize=14)
    plt.ylabel('True', fontsize=14)
    plt.show()

def model_metrics(a, b):

    accuracy = metrics.accuracy_score(a, b)
    precision = precision_score(a, b)
    recall = recall_score(a, b)
    f1 = f1_score(a, b)

    print('Accuracy:', round(accuracy*100, 2),'%')
    print('Precision score:', round(precision*100, 2),'%')
    print('Recall score:', round(recall*100, 2),'%')
    print('F1 score:', round(f1*100, 2),'%')

# Target (y) and Features (X) definitions:
```

```
y = df['ARR_FLIGHT_STATUS']
X = df.drop(['ARR_FLIGHT_STATUS'], axis=1)


df.head(2)

input_shape_column=len(df.columns)-1

input_shape_column


# # Neural Network Conditions
# Data has to be purely numerical
#
# Data cannot contain missing values
#
# Data has to be Normalized


df.isna().sum().sum()


# We know the data is purely numerical and that it has no missing values, now all is needed is
to normalized and we will do it by using the StandardScaler

col_names = list(df.columns)

s_scaler = preprocessing.StandardScaler()
df_s = s_scaler.fit_transform(df)

df_s = pd.DataFrame(df_s, columns=col_names)


# The following are the two first rows of the normalized data:


df_s.head(2)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)


# # Modeling

# # Model MLP NN
```

```
model_MLP_NN= Sequential()

model_MLP_NN.add(Dense(30, activation='relu', input_shape=(input_shape_column,)))

model_MLP_NN.add(Dense(10, activation='relu'))

model_MLP_NN.add(Dense(5, activation='relu'))

model_MLP_NN.add(Dense(1, activation='sigmoid'))

model_MLP_NN.summary()

model_MLP_NN.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

results_MLP_NN = model_MLP_NN.fit(X_train, y_train, epochs=25, batch_size=32,
validation_split=0.1)



visualize_training_results(results_MLP_NN)

y_pred_MLP_NN = model_MLP_NN.predict(X_test)
y_pred_m7 =(y_pred_MLP_NN > 0.5)

cm_MLP_NN = confusion_matrix(y_test, y_pred_MLP_NN)
print(cm_MLP_NN)
print("------------------")
print(classification_report(y_test, y_pred_MLP_NN))

conf_matrix(cm_MLP_NN)

model_metrics(y_test, y_pred_MLP_NN)
```