Flight Status Prediction: A Comparative Analysis

Ji Hwan Kim (jhklc21), Madhu Sree Sankaran (mss1g22), Maria Elkjær Montgomery (mem1d22), Samyuktha Babuganesh (sb9n22), and Saran Raj Rajasekaran (srr1g21)

ABSTRACT

Flight delays are a common occurrence in the real-world, impacting the revenue of airline companies, due to delay compensation, resulting in lower customer satisfaction. It also has a significant impact on the consumer's normal travel plans. Thus, predicting flight delays is an important application. In this assignment we experimented with various Machine Learning and Deep Learning approaches with the goal of predicting arrival flight delays using the 2022 data from the Flight Status Prediction dataset [12]. The results of the approaches were then recorded and compared. Multinomial Logistic Regression, K-Nearest Neighbour, Random Forest, Ada Boost and Neural Networks were used for multi-class classification and the results show that the Neural Network performed the best with an accuracy of 79.63% on an imbalanced dataset and 73.46% on a balanced dataset.

1 INTRODUCTION

Airline delays are an important topic of concern for the aviation industry. Since the impact of the global pandemic has now waned, more people are turning to airlines as their preferred mode of transportation. "In 2022, air passenger traffic gained momentum globally and recovered substantially from 41.7% of 2019 revenue passenger-kilometers (RPKs) in 2021 to 68.5% in 2022" [11]. However, as the demand increases, so do the delays. The Federal Aviation Administration (FAA) [10], estimated that the cost of delays had increased from $23.7 billion dollars in 2016 to $33.0 billion dollars in 2019. Moreover, nearly half the cost of delays were the cost incurred to compensate passengers for the delayed flights. Flight delays cost airlines millions of dollars per year, while causing a great inconvenience to the passengers [8]. There is a clear incentive for the aviation industry to prevent or minimise delays. Thus, predicting flight delays is an important application for airline companies to reduce the incurred cost and improve customer satisfaction. This is also useful to passengers, allowing them to make plans based on approximations of flight delays.

The chosen dataset was the Flight Status Prediction [12] dataset from Kaggle with information being collected from the Bureau of Transportation Statistics (BTS) in the United States. The dataset contains information related to aviation delays for Domestic airlines within the USA from the year 2018 to 2022. Flight delays can be categorised into 3 types, namely: Small delays (5 - 10 minutes), Medium delays (15 - 45 minutes) and Large delays (more than 45 minutes) [16]. Thus, a multi-class classification problem was implemented with 5 classes related to arrival delays and cancellations with the following labels: Small delay, Medium delay, Large delay, Cancelled and On-time. Moreover, flight delays for the year 2022 was chosen and the multi-class classification problem was implemented using 4 Machine Learning approaches (Logistic Regression, K-Nearest Neighbour, Random Forest and Ada Boost) and 1 deep leaning approach (Neural Network).

2 DATA ANALYSIS

2.1 Data Source

The Flight Status Prediction [12] dataset on Kaggle consists of separate CSV files, with a total of 5 such files, one for each year from 2018 to 2022. The files for each year contain millions of training instances and so a single file, for the year 2022 was used. This file consists of 61 features related to flight delays and cancellation, with around 4 million training instances for the months January to July of 2022.

2.2 Data Pre-processing

The features of the dataset were first manually inspected in order to remove those that were indicative of the classification problem. These features include information related to cancellation, delay time for arrival and departures, etc. There were also several features in the dataset that represented the same information in different formats. These repetitive features were manually removed such that only a single representative feature for the delay and cancellation was included. This manual inspection led to the removal of 29 features, bringing down the total number of features to 32. The list of features, a description and the reasons for being dropped is shown in the appendix.

The features in the dataset were both categorical and numerical, and so the Label Encoder was used to convert the categorical features into numeric variables. One hot encoding was also considered and implemented, but due to the high cardinality in each of the features, this led to a high-dimensional vector being created for each feature, further leading to an increase in the amount of memory and Out-of-Memory issues. "This high dimensionality entails large computational and memory costs; it increases the complexity of the associated learning problem, resulting in a poor statistical estimation" [2]. Thus, the sparse representation of the features, owing to the high cardinality and dimensionality was avoided and a simple label encoder was used. The predictive label was encoded using one-hot vector encoding.

Lastly, the features containing time were originally encoded in the integer format 'hhmm', e.g. 5.30PM would be written as '1730'. These numbers were converted to floats referencing the time in hours, such that 1730 was converted to 17.5.

The dataset was then split into a training (70%), validation (20%) and test set (10%).

2.3 Exploratory Data Analysis

Data Visualisation was performed in order to understand the data and its characteristics.

The distribution of delay ranging from 1 to 60 minutes and the class distribution is shown in figure 1. From these plots, we can infer that most of the flights in the year 2022 were on time, with few cancellations. Moreover, small delays (5 to 15 minutes) appear to be more prevalent than large delays.

, ,

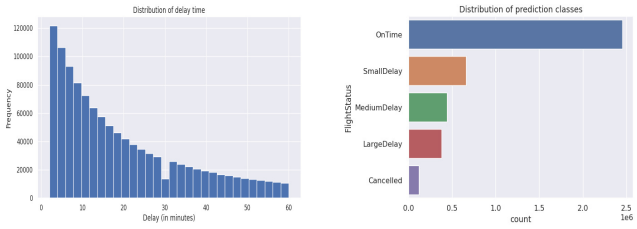

Figure 1: Delay and Class Distribution

Figure 2 shows all the different airline companies that are in the dataset, and the distribution of flight status for each company. Here we can see that the dataset includes a total of 22 companies, and that the distribution of the different predictive classes are somewhat similar for each company. There are however some that have a higher share of delays than others.

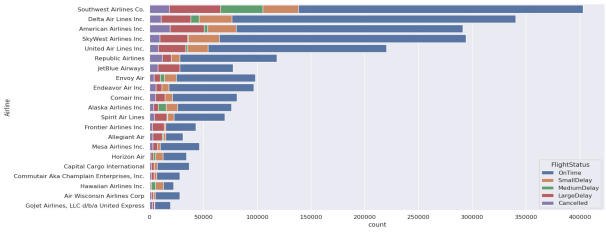

Figure 2: The different airlines and the delays

Figure 3 shows all the different planned departure times for the flights in the dataset in 24-hour time. Here we see that there are no flights between 2.00 and 4.00, and that the majority of the flights are between 06.00 and 07.00. It is also evident that the times with the most large delays and cancellations are from 17.00-19.00.

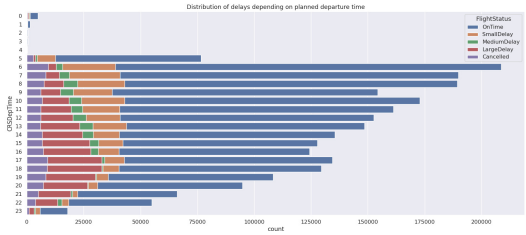

Figure 3: The predictive classes in the different planned departure times

Finally, figure 4 shows a heatmap over the months and the delays. As the figure shows we only have data from the months January to July. Additionally we can also see that January has the least amount of delays, but a high number of cancelled flights, whereas the months from March to July have a higher percentage of delays.

2.4 Balancing the dataset

Figure 1 shows the imbalance in the dataset, where there are more training instances for on-time arrivals, when compared to the other

FlightStatus	OnTime	SmallDelay	MediumDelay	LargeDelay	Cancelled
Month					
1	62.480335	13.523156	9.210023	8.446373	6.340112
2	62.295404	15.092704	9.952377	8.147425	4.512091
3	61.175328	16.882123	11.227190	9.169169	1.546190
4	59.503920	17.362471	11.243086	9.576298	2.314225
5	60.538870	17.574900	11.045132	8.846815	1.994283
6	57.402771	17.068376	11.782765	10.668839	3.077249
7	59.472179	16.386961	11.566732	10.769278	1.804850

Figure 4: Heatmap over delays for each months

classes. There are 2 different methods to balance the dataset, namely: Oversampling and Undersampling. The Oversampling method creates synthetic samples of the data for the under-represented classes, whereas the undersampling method only considers a random subset of the the majority class to balance the dataset. The imbalanced training set was balanced using undersampling. Undersampling was chosen to prevent overfitting, and to minimise the amount of data to train on, as the imbalanced dataset is very large.

3 FEATURE SELECTION AND EXTRACTION

Feature Selection and Feature Extraction is not only used to reduce the dimensionality of the dataset but also allows the Machine Learning model to be trained on the most relevant information for the given task. To ensure that the right model was chosen for feature selection, the relationship between the features were determined using the Pearson's Correlation as shown in figure 5.

Pearson's Correlation Coefficient is a measure of strength and direction of the linear relationship between features/variables. Coefficient values of 1 indicate a strong linear relationship and a value of 0 indicates that there is no linear relationship. The correlation heatmap shown in 5, indicates that most features are uncorrelated, with a correlation coefficient of 0 or have a weak positive/negative correlation, with values in the range of [-0.2, 0.2]. This heatmap does indicate that the variables may not have a linear relationship.

Due to the possible non-linear nature of the data, Principle Component Analysis (PCA) was not chosen. This is due the assumption made by PCA, that the relationship between the features are linear, and thus PCA will be unable to capture the underlying complexity of the data. Kernel PCA, an extension of PCA that works well when the relationship between the features are non-linear, was also considered and implemented. However, the major disadvantage of kernel PCA is the construction of the kernel matrix, which consists of N row and N columns, where N is the number of training instances. "This is an unfortunate limitation as in practice, to obtain the best model, we would like to estimate the kernel principal components from as much data as possible" [15]. Thus, decision trees and Auto-Encoders were used to find the feature importance and perform feature selection. These methods were chosen as they have shown good results for dimensionality reduction of non-linear data [3, 14].

For the the decision tree dimensionality reduction, a decision tree was implemented on the training data, using `sklearn`'s standard implementation. The most important features from the trained decision tree were then extracted. For the autoencoder a basic encoder using `keras` library was implemented with an encoding

dimension of $n = 10$. This was implemented very similarly to the basic implementation shown on the keras website [4]. The encoder consisted of one dense layer, using the ReLU activation function, and the decoder was implemented as a dense layer using the Sigmoid activation function. The autoencoder was then trained to encode and decode the training data with the Adam optimiser, and a crossentropy loss. The highest weighted features were then extracted.

Finally, the highest weighted features from both the dimensionality reduction methods were compared and selected as features. These where the top 14 features from both methods, respectively. These include the following: Airline, Origin, Dest, Diverted, CRSDepTime, DepTime, CRSElapsedTime, Distance, Quarter, Month, DayOfMonth, Marketing_Airline_Network, and Operated_or_Branded_Code_Share_Partners.

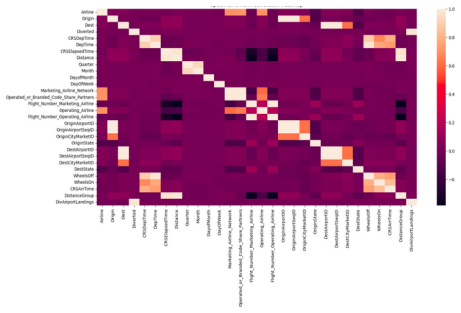

Figure 5: Pearson correlation matrix for the features

4 MACHINE LEARNING MODELS

The 4 classical Machine Learning approaches, namely: Multinomial Logistic Regression, K-Nearest Neighbour, Random Forest and AdaBoost were implemented using the sklearn library. The neural network was implemented using the Keras library.

4.1 Multinomial Logistic Regression

One of the simplest and most prominent algorithms used for classification is Logistic Regression, which is a probabilistic model. Traditionally, this algorithm is used for Binary Classification, where there are only 2 classes for the dependant variable. However this can be extended to represent a multi-class classification problem and is known as Multinomial Logistic Regression. This algorithm can be used to find the probability distribution over multiple classes rather than just 2 classes and will return the class prediction that has the highest probability. This has been used as a baseline model to compare and evaluate the performance with all the other models.

The model was tuned, trained and run for a total of 1000 iterations with the L2 regularisation to prevent overfitting.

4.2 K-Nearest Neighbour (KNN)

KNN is an instance based, non-parametric model that does not make any assumptions about the underlying distribution of the data. The KNN algorithm is also known as a lazy learner as it does not learn any information during the training phase. When making predictions, the model will find the K closets training data points, based on the distance metric. There are several distance metrics that can used to compute the distance, namely: Manhattan distance, Euclidean distance, cosine distance, cityblock distance, etc. The model is however, highly dependant on hyper parameter tuning for choosing the value of K and the distance metric. The KNN model was chosen for this problem, as multiple similar studies focusing on the prediction of flight delays have shown favourable results using a KNN [6, 7].

The model was trained and tuned on the validation set and 11 neighbours were chosen, as higher values led to overfitting. Moreover, the Manhattan distance was chosen for the distance metric that the model was trained on.

4.3 Random Forest

Random Forest is an ensemble learning method that uses an ensemble of decision trees to train the model. It uses the concept of bagging, where each decision trees is trained in parallel on a random subset of the training set (with replacement). The output of the classifier will be the class that receives the majority vote from each individual decision trees. Random Forest also adds more randomness to the model by choosing the best split for the decision trees, from a random subset of features. This results in a model that has a lower variance, and thus generalises better on new unseen instances.

Thiagarajan.B, et al.[8], proposed a 2-stage model to predict the flight delay in minutes, where the first stage was a binary classification of both departure and arrival departure delays. In the first stage, the Random Forest Algorithm gave the second highest performance when predicting arrival delays, with an accuracy of 94.09%, using a set of on-time performance and weather features. Thus, Random Forest was chosen as one of the classifiers for implementation.

The model was trained and tuned on the validation set by performing a grid search on the number of estimators, the maximum depth of the trees, the maximum features, and the minimum sample splits. From this the final model was implemented with 100 estimators, a maximum depth of 10, the square root for computing the total number of features, and a 5 minimum sample split.

4.4 AdaBoost

AdaBoost, or Adaptive Boosting, is a boosting/ensemble algorithm which utilises multiple weak learners to create one combined strong classifier. This is done by iteratively training the weak learners on a training dataset, where each new weak learner focuses on the data that the previous learners misclassified. This focus on the misclassified data from the previous weak learners, is what makes the algorithm adaptive. The final output of the classifier is a weighted output of the all the weak learners [9]. Typically, AdaBoosting utilises weak base learners; decision stumps being the most commonly used base estimator. There are however some results that have shown that the algorithm can be successful when utilising decision trees, such that the method works similar to that of a random forest [5]. For the hyperparameter tuning of this model we chose to focus on the number and type of weak learners the model employs. To pick the optimal parameters a gridsearch was performed using $n = [10, 50, 100, 200]$ estimators, and base learners consisting of decision trees with the maximum depth of

$d = [1, 2, 3, 4, 5]$. From the gridsearch all the different variations of those hyperparameters were trained on the training set, and then tested on the validation set. The model which achieved the highest F1-score on the validation set was then picked as the final model, after which its performance was tested on the test data. This was done for both the imbalanced and the balanced data.
The performance of the models on the validation set are shown in figure 6. Here we see that the model trained on the imbalanced dataset has a much higher accuracy than the one trained on the balanced dataset, and that both models have similar F1 scores. From these runs $n = 200$ and $d = 2$ for the balanced dataset, and $n = 200$ and $d = 5$ for the imbalanced dataset were chosen as the hyperparameters.

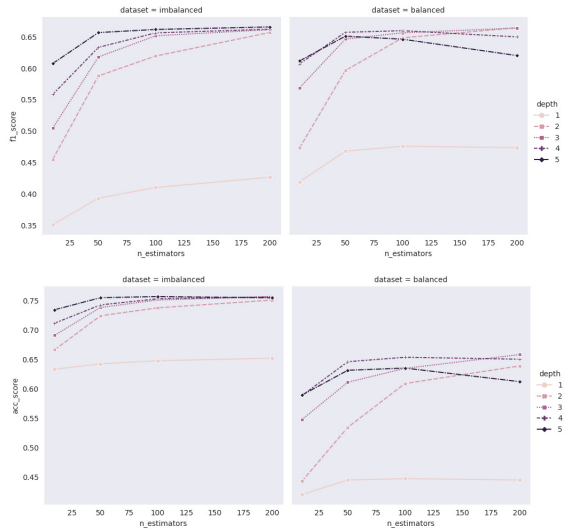

Figure 6: F1 scores (top) and accuracy scores (bottom) for the different AdaBoost models. Left = imbalanced, right=balanced

4.5 Neural Network

A neural network is an algorithm that models complex patterns and relationships in data. The simplest architecture of this algorithm is a multi-layer perceptron (MLP). It usually consists of fully connected layers, where all neurons in a layer are connected to every neuron in the next one. In this network, the information will be propagated to the endpoint, and the network will be updated based on the outcome of this information transmission.

Hajar Alla, et al. [1], proposed a Multi-layer perceptron (MLP) to predict arrival flight delays. The authors used 2 hidden layers with 50 units each, the ReLU activation function and the Adam Optimiser. They were able to show that this deep learning based approach was able to outperform Gradient Boosting and Decision trees with an accuracy of 90.48%. Using the approach of selective data training, to deal with overfitting, they were able to show an improved accuracy of 95.60%, thus motivating the use of Neural networks for this application.

The neural network has many hyperparameters to be set. The number of layers and that of nodes per layer are the basic hyperparameters to tune. A more significant number of them can make the model complex, fundamentally resulting in better performance. Another hyperparameter is the activation function, where the typical activation functions are ReLU or Sigmoid. The learning rate, optimisation function, batch size, the number of training epochs and regularisation techniques are other examples of hyperparameters.

The multi-layer perceptron is used for this coursework. The model was first trained on an imbalanced dataset and the training, validation loss and accuracy are shown in 7. This model used three hidden layers, with 256 nodes for the first two layers and 50 neurons for the last layer. In addition, the model was trained for 50 epochs with the Adam optimiser. Under this condition, a set of different batch sizes and learning rates were used to train the model and evaluation was done on the validation set. As a result, the batch size of 256 and the learning rate of 0.0005 were chosen since they showed the best overall results.

Similarly, another MLP model with a balanced dataset used three hidden layers, with 512, 256, and 50 nodes, respectively. The training, validation loss and accuracy on the balanced dataset are shown in 7. The model was trained for 50 epochs with the Adam Optimiser. After tuning the hyperparameter of its batch size and learning rate, the size of 64 and the rate of 0.0001 were chosen.

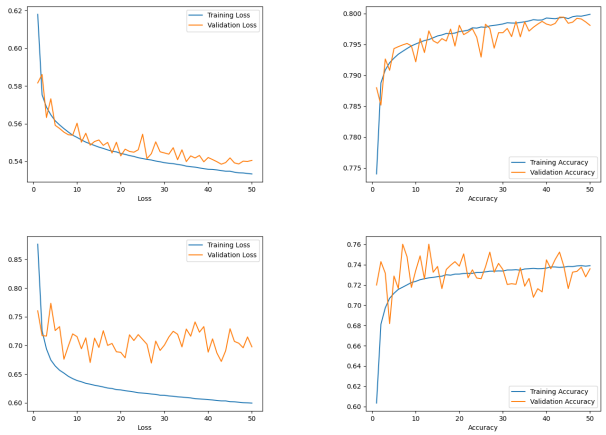

Figure 7: Training, Validation Loss and Accuracy for imbalance (Top) and balance (bottom)

5 PERFORMANCE

The models were trained on the imbalanced dataset and then trained on a balanced dataset. The following evaluations metrics were used, namely: Accuracy, Macro Averaged Precision, Macro Averaged Recall and Macro Averaged F1 Score.
The performance of all the models on the imbalanced dataset is shown in table 1. The performance of all the models on the balanced dataset is shown in table 2.

6 DISCUSSION

Based on the above results, we can see that, overall the Neural Network performs the best for both the datasets, followed by AdaBoost, then Random Forest, and finally the K-Nearest Neighbour and the Logistic Regression. It is also clear that the models trained on the imbalanced dataset outperform the models trained on the balanced

Models	Accuracy	Precision	Recall	F1 Score
Multinomial Logistic Regression	65.82%	65.15%	45.27%	43.57%
K-Nearest Neighbour	64.30%	56.35%	46.46%	47.62%
Random Forest	64.54%	72.24%	64.54%	52.02%
Ada Boost	75.32%	70.02%	65.96%	66.45%
Neural Network	79.63%	80.14%	71.53%	74.66%

Table 1: Imbalanced Dataset

Models	Accuracy	Precision	Recall	F1 Score
Multinomial Logistic Regression	46.06%	45.42%	52.77%	47.16%
K-Nearest Neighbour	61.90%	52.33%	36.71%	40.31%
Random Forest	40.68%	44.48%	37.59%	40.48%
Ada Boost	63.87%	67.07%	66.77%	66.48%
Neural Network	73.46%	74.98%	73.34%	73.87%

Table 2: Balanced Dataset

dataset, especially in the accuracy scores. This was expected for the accuracy scores, but we expected that the models trained on the balanced dataset would outperform on the F1-scores, since the raw dataset was very imbalanced. The reason behind these results may be that the undersampling method used to obtain the balanced dataset, resulted in removal of too much data. Thiagarajan.B, et al.[8], implemented a binary classification of flight delay predictions and found that oversampling produced better results than undersampling using Random Forest, on the dataset. "Undersampling considers only a sample from the majority class thus neglecting potentially important information in the ignored examples albeit the fact that it significantly reduces the run time" [8]. For future work, it may therefore be relevant to try other sampling methods, such as oversampling, to mitigate these problems. This would however also affect the run-time of the model training, which was already substantial.

Moreover, the results support the notion that the data may be non-linear as methods such as the logistic regression performed so poorly compared to the other methods. From this perspective the performance of the Random Forest is however surprising, as this model tends to perform well in other similar studies[8]. This discrepancy could be due to Random Forest model's tendency to overfit when subjected to data that is too noisy [13], or simply because the resampled data was too sparse.

7 CONCLUSION

Flight delay prediction and classification is a relevant issue to most passengers and airlines. Prior information related to delays in the scheduled flights has the potential to allow passengers to have knowledge regarding changes in scheduling and also allows the airline companies to improve their customer service. Classification of arrival delays was implemented successfully using Machine Learning and Deep learning based approaches. This application can also be extended to predicting departure delays. The neural network had the best overall accuracy of 79.63% and 73.46% on both the imbalanced and balanced dataset. Moreover, all of the features except for one (the departure time) that the model was trained on are features that are available before the departure of the flights. This suggests that a Deep Learning approach similar to the one implemented, could be useful in the prediction of delays before the actual flight's departure.

Some of the limitations of the research include the fact that most of the information is from domestic flights within the United States, from the months of January to July of 2022. This can be further extended to include information regarding international delays. Moreover, the features do not include any weather related information, which maybe useful to provide reasoning behind flight delays, particularly for longer international flights. For future work, it may therefore be of interest to improve upon these limitations.

REFERENCES

[1] Hajar Alla, Lahcen Moumoun, and Youssef Balouki. 2021. A Multilayer Perceptron Neural Network with Selective-Data Training for Flight Arrival Delay Prediction. *Sci. Program.* 2021 (2021), 5558918:1–5558918:12.

[2] P. Cerda and G Varoquaux. 2022. Encoding High-Cardinality String Categorical Variables. *IEEE Transactions on Knowledge and Data Engineering, 34(3)* (2022). https://doi.org/10.1109/TKDE.2020.2992529

[3] Jung Hwan Cho. 2011. "Decision tree approach for classification and dimensionality reduction of electronic nose data". *Sensors and Actuators B: Chemical* (2011). https://doi.org/10.1016/j.snb.2011.08.027.

[4] Francois Chollet. 2016. *Building Autoencoder in Keras.* https://blog.keras.io/building-autoencoders-in-keras.html

[5] Abraham J. Wyner et. al. 2017. Explaining the Success of AdaBoost and Random Forests as Interpolating Classifiers. *Journal of Machine Learning Research* (2017). https://jmlr.org/papers/volume18/15-240/15-240.pdf

[6] Blessy Trencia Lincy et. al. 2022. "Analysis of Flight Delay Data Using Different Machine Learning Algorithms". *New Trends in Civil Aviation* (2022). https://doi.org/10.23919/NTCA55899.2022.9934398

[7] Jia Yi et. al. 2021. "Flight Delay Classification Prediction Based on Stacking Algorithm". *Journal of Advanced Transportation* (2021). https://doi.org/10.1155/2021/4292778

[8] Thiagarajan B et al. 2017. "A Machine Learning Approach for Prediction of On-Time Performance of Flights". *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, (September 2017). https://doi.org/10.1109/DASC.2017.8102138

[9] Yoav Freund et. al. 1999. A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence,*. (September 1999). https://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf

[10] Federal Aviation Administration (FAA). 2019. Cost of Delay Estimates. (2019). https://www.faa.gov/sites/faa.gov/files/data_research/aviation_data_statistics/cost_delay_estimates.pdf

[11] International Air Transport Association (IATA). 2022. Air Passenger Market Analysis. (2022). https://www.iata.org/en/iata-repository/publications/economic-reports/air-passenger-market-analysis---december-2022/

[12] Rob Mulla. 2013. *Flight Status Prediction.* https://www.kaggle.com/datasets/robikscube/flight-delay-dataset-20182022

[13] Rebellion Research. 2022. *What are the disadvantages of random forest?* https://www.rebellionresearch.com/what-are-the-disadvantages-of-random-forest

[14] CV. Sugumaran. 2007. "Feature selection using Decision Tree and classification through Proximal Support Vector Machine for fault diagnostics of roller bearing". *Mechanical Systems and Signal Processing*, (2007). https://doi.org/10.1016/j.ymssp.2006.05.004.

[15] Michael Tipping. 2000. Sparse Kernel Principal Component Analysis. (2000). https://proceedings.neurips.cc/paper_files/paper/2000/file/bf201d5407a6509fa536afc4b380577e-Paper.pdf

[16] Wikipedia. 2023. *Flight Cancellation and Delay.* https://en.wikipedia.org/wiki/Flight_cancellation_and_delay

A FEATURES

Feature	Description	Type	Contains NaN	Conversion	Dropped
'FlightDate'	Date of flight	String	0		Dropped due to repetition
'Airline'	Which airline is flying the plane	Categorical values	0		
'Origin'	City name of flight origin	Categorical values	0		
'Dest'	City name of flight destination	Categorical values	0		
'Cancelled'	Whether flight is cancelled or not	Binary	0		Dropped as it is indicative
'Diverted'	Whether flight is diverted or not	Binary	0		
'CRSDepTime'	Scheduled time of departure	Float (hhmm)	0	Converted to 'time float'	
'DepTime'	Actual departure time	Float (hhmm)	1	Converted to 'time float'	
'DepDelayMinutes'	Departure delay in minues	Float	1		Dropped as it is indicative
'DepDelay'	Same as DepDelayMinutes with different way of handling early flights	Float	0		Dropped as it is indicative
'ArrTime'	Actual arrival time	Float (hhmm)	1		Dropped as it is indicative
'ArrDelayMinutes'	Array delay in minutes	Float	1		Dropped as it is indicative
'AirTime'	Actual time plane is in the air in minutes	Float	1		Dropped as it is indicative
'CRSElapsedTime'	Plane time for flight to be in air in minutes	Float	0		
'ActualElapsedTime'			1		Dropped as it is indicative
'Distance'	Distance between airports in miles	Float	0		
'Year'	Year	Int	0		Dropped as it only contains 2022
'Quarter'	Quarter of the year	Int	0		
'Month'	Which month flight is in in numbers	Int	0		
'DayofMonth'	Which day in the month flight departs	Int	0		
'DayOfWeek'	Which day of the week flight is in	Int	0		
'Marketing_Airline_Network'	Marketing airline	Categorical values	0		
'Operated_or_Branded_Code_Share_Partners'	Operational partnership	Categorical values	0		
'DOT_ID_Marketing_Airline'	ID for 'Marketing_Airline_Network'	Categorical values	0		Dropped due to repetition
'IATA_Code_Marketing_Airline'	ID for Operated_or_Branded_Code_Share_Partners	Categorical values	0		Dropped due to repetition
'Flight_Number_Marketing_Airline'	Flight number for marketing airline	Categorical values	0		
'Operating_Airline'	Which airline is operating the flight	Categorical values	0		
'DOT_ID_Operating_Airline'	Same as 'Operating_Airline'	Categorical values	0		Dropped due to repetition
'IATA_Code_Operating_Airline'	Similar to 'DOT_ID_Operating_Airline'	Categorical values	0		Dropped due to repetition
'Tail_Number'	Number of tail on the flight	Categorical values	0		
'Flight_Number_Operating_Airline'	Flight number for operating airline	Int	0		
'OriginAirportID'	ID for origin airport	Int	0		
'OriginAirportSeqID'	Same as 'OriginAirportID' but with coordinated added	Int	0		
'OriginCityMarketID'	???	Int	0		
'OriginCityName'	Contains city name and state	Categorical values	0		Dropped due to repetition
'OriginState'	Acronym for origin state	Categorical values	0		
'OriginStateFips'	Same as 'OriginState'	Categorical values	0		Dropped due to repetition
'OriginStateName'	Same as 'OriginState'	Categorical values	0		Dropped due to repetition
'OriginWac'	Same as 'OriginState'	Categorical values	0		Dropped due to repetition
'DestAirportID'	ID for desitnation airport	Int	0		
'DestAirportSeqID'	Same as 'DestAirportID' but with coordinated added	Int	0		
'DestCityMarketID'	???	Int	0		
'DestCityName'	Contains city name and state	Categorical values	0		Dropped due to repetition
'DestState'	Acronym for origin state	Categorical values	0		
'DestStateFips'	Same as 'DestState'	Categorical values	0		Dropped due to repetition
'DestStateName'	Same as 'DestState'	Categorical values	0		Dropped due to repetition
'DestWac'	Same as 'DestState'	Categorical values	0		Dropped due to repetition
'DepDel15'	Binary value that indicates if flight is delayed by more than 15 minutes	Binary	0		Dropped due to repetition
'DepartureDelayGroups'	Groups that indicate how much the plan is delayed	Categorical values	0		Dropped due to repetition
'DepTimeBlk'	Departure time block	Categorical values	0		Dropped due to repetition
'TaxiOut'	Time to taxi out in minutes	Float	0		Dropped as it is indicative
'WheelsOff'	Time when wheels are off (local time)	Float (hhmm)	1	Converted to 'time float'	
'WheelsOn'	Time when wheels are on (local time)	Float (hhmm)	1	Converted to 'time float'	
'TaxiIn'	Time to taxi in minutes	Float	1		Dropped as it is indicative
'CRSArrTime'	Time when flight should arrive	Float (hhmm)	0	Converted to 'time float'	
'ArrDelay'	Delay in arrival	Float	1		Dropped as it is indicative
'ArrDel15'	Binary value that indicates if flight is delayed by more than 15 minutes	Binary	0		Dropped as it is indicative
'ArrivalDelayGroups'	Groups that indicate how much the plane is delayed	Categorical values	0		Dropped as it is indicative
'ArrTimeBlk'	Arrival time block	Categorical values	0		Dropped due to repetition
'DistanceGroup'	Distance intervals	Categorical values	0		
'DivAirportLandings'	Number of diverted airport landings	Int	0		