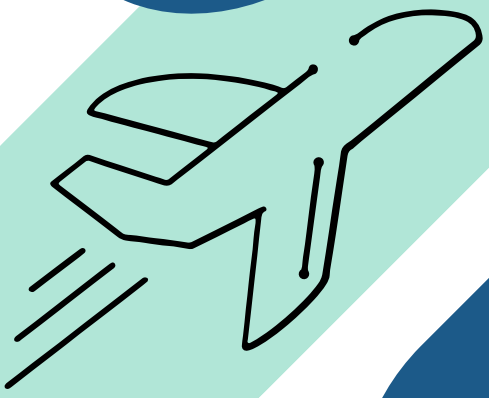
Exam Paper with Oral Exam
Second Semester, Spring 2023



PREDICTING

Flight Delays

BSc in Business Administration and Digital Management
Copenhagen Business School

Course: Business Data Analytics, Quantitative Methods and Visualisation

Instructors: Daniel Hardt and Chris Zimmerman

Submission date: 24th May

Number of characters: 32,201

Number of physical pages: 15

Authors:

Jasmin El Desouky (161273)

Ana Rebelo da Fonseca (161587)

Mathias Havshov Artmann (162165)

Table of contents

Introduction

In today's world, the aviation industry holds a significant position in promoting global connectivity and driving economic growth. It can be considered one of the most competitive industries, contributing a sum of $2.7 trillion (3.6 per cent) to the world's GDP (Asquith, 2020). However, despite its importance, the industry faces challenges known to disrupt its operation. One such challenge is flight delays, which can not only impact the industry's efficiency and profitability but can also cause a significant inconvenience to many passengers. As one once said, "time is money" and therefore every minute of delay is a direct hit on an airline's profitability.

Subsequently, this paper will utilize a dataset that includes information regarding flight delays and non-delays, complemented by accompanying characteristics. The aim is to perform a range of machine learning models, improving each one of them in the "train" stage, thereby obtaining the best possible result when "testing." Predicting flight delays successfully would allow firms to improve their operations, resulting in more satisfied customers and greater profits. This has the potential to revolutionize the aviation industry positively for many years to come.

Business Relevance

Delays are one of the many challenges encountered in the aviation industry significantly disrupting its seamless operation. In 2019, a total of 1.45 million flight delays were experienced by US airlines, resulting in a cost of $1.1 billion. Similarly, in Europe, 19.1 million minutes of delay results in costs of €9.6 billion (Vijayan, 2023). As a result, to address the challenges, which flight delay poses, this paper will leverage various machine learning and data modeling techniques to predict whether a flight will get delayed.

If proven to be successful, this would aid many airlines globally to mitigate flight delays, thereby improving not only profitability but also operational efficiency, resource optimization and customer satisfaction. Delays can be a huge financial cost to many airlines including the cost of fuel, aircraft maintenance staff, and crew overtime, therefore accurately predicting flight delays, allows airlines to reduce these costs (Khan et al., 2022). In addition to this, the huge cost burden of refunding tickets and fines for every minute an aircraft is parked after estimated departure can significantly affect profitability. Therefore, all can be prevented with accurate predictions on flight delays allowing one to optimize the allocation of resources and prevent such occurrences. Moreover, flight delays often cause many distressed and unsatisfied customers (Efthymiou et al., 2019), so by utilizing machine learning we can discover the underlying factors. As a result, it will allow airlines to adopt measures such as providing up-to-date timely updates or potentially offering alternative travel options. Improving customer satisfaction would not only improve an airline's brand image, but also customer loyalty. Furthermore, flight delays can cause a cascade of effects not only on other departing flights but as well other industries including ground handling services, logistics companies, and other stakeholders. On-time flights would result in greater efficiency and cost savings, where for example ground handling

services can optimally adjust staff deployment in accordance with predicted departure times. Finally, research has shown that flight delays can significantly impact the environment, as it emits additional CO2 than prognosed due to additional aircraft fuel being burned every time it is parked after estimated departure (Dissanayaka et al., 2019). As a result, accurate predictions enable airlines to optimize time schedules and fuel usage more effectively.

Regardless of the multiple benefits that adopting a machine learning algorithm to predict flight delays poses. One must also consider the possible societal or ethical issues. One of which is privacy and data security. This is because accurately predicting flight delays may require access to a significantly large amount of sensitive operational data. Therefore, it is important to handle this data responsibly and prevent any misuse or data leaks. Furthermore, one must ensure that the model developed to predict flight delays is bias-free, taking into consideration numerous factors that may affect delays. Predicting flight delays could potentially cause ethical issues if customer data is incorporated into the dataset such as race, ethnicity and gender, there is a risk that the model developed may discriminate against various groups assuming that there is causation rather than just being a case of correlation. This is why airlines should also be aware of the ethical implications that arise with such predictions. Particularly, ensuring that the use of delay predictions is not utilized to unfairly prioritize services or discriminate against passenger groups based on expected delays. It is critical to treat all passengers equally, regardless of the flight status and to guarantee that they all receive a similar experience.

Predicting flight delays using machine learning has the potential to completely transform the aviation sector by providing passengers throughout the world with a more comfortable and reliable flying experience. However, this can only be successful when there is a right balance met between business objectives and ethical issues.

Data Description

The objective of this research is to be able to predict based on numerous factors whether a flight is being delayed more than 15 minutes or not (*Types of Delay*, n.d.). It would be almost impossible to take-off on the planned minutes every time, and therefore a benchmark is needed for when the plane is considered delayed. We decided to follow the guidelines of the Federal Aviation Administration of US (FAA), which considers a flight delayed if the delay exceeds 15 minutes from planned departure time.

The extended Flight Status Prediction dataset originally has 44 columns and 6,489,064 rows (*2019 Airline Delays W/Weather and Airport Detail*, 2022). Of which 5,261,694 are flights that are not delayed and 1,227,368 are delayed, comprising 18.9%. We are using a customized version of a Kaggle-dataset that includes extensive flight data from 2019 combined with data from other sources including weather and company information. Both numerical and categorical data are present in the data.

Feature	Description
MONTH	Month 1-12
DAY_OF_WEEK	Day of Week 1-7
TAIL_NUM	Unique Aircraft Identifier
DEST_AIRPORT_ID	Destination Airport (ID)
DEP_DEL15	Target: Binary Classification (1 = Plane more than 15 minutes late, 0 is under)
DEP_TIME_BLK	Departure Time Block
ARR_TIME_BLK	Arrival Time Block
DISTANCE_GROUP	Distance Group
CONCURRENT_FLIGHTS	Concurrent flights leaving from the airport in the same departure block
NUMBER_OF_SEATS	Number of seats on aircraft
CARRIER_NAME	Carrier Name
AIRPORT_FLIGHTS_MONTH	Average Airport Flights per Month
AIRLINE_FLIGHTS_MONTH	Average Airlines Flight per Month
AIRLINE_AIRPORT_FLIGHTS_MONTH	Average Airlines Flight per Month for that Airport
AVG_MONTHLY_PASS_AIRPORT	Average amount of passengers at departure airport a month
AVG_MONTHLY_PASS_AIRLINE	Average amount of passengers for airline per month
FLT_ATTENDANTS_PER_PASS	Flight Attendants per passenger for airline
GROUND_SERV_PER_PASS	Ground service employees (service desk) per passenger for airline
PLANE_AGE	Plane Age in Years
DEPARTING_AIRPORT	Departing Airport Name
PREVIOUS_AIRPORT	The Airport that flight was before if any
PRCP	Inches of precipitation for day
SNOW	Inches of snowfall for day
SNWD	Inches of snow on ground for day
TMAX	Max temperature for day
AWND	Max wind speed for day

Figure 1 Feature Descriptions

We have decided to remove a selection of the columns/features from the original dataset that could potentially be highly correlated and could risk exposing the result to machine learning algorithms. Some of this information will also first be attainable after a flight has occurred and will therefore have no relevance in the prediction of future flights. These columns include: 'DAY_OF_MONTH', 'CRS_DEP_TIME', 'DEP_TIME', 'DEP_DELAY_NEW', 'CRS_ARR_TIME', 'ARR_TIME', 'ARR_DELAY_NEW', 'CRS_ELAPSED_TIME', 'ACTUAL_ELAPSED_TIME', 'DISTANCE', 'LATITUDE', 'LONGITUDE', 'CARRIER_DELAY', 'SECURITY_DELAY', 'WEATHER_DELAY', 'NAS_DELAY', 'LATE_AIRCRAFT_DELAY', 'SEGMENT_NUMBER'.

The columns that are kept can be seen in figure 1 together with the description. There are a total of 26 columns left, where 25 are features and 1 is the target. We will be doing a binary classification based on the column DEP_DEL15, where 1 denotes a flight delayed more than 15 minutes and 0 denotes a flight that was not delayed.

The remaining features that were kept in the data are expected to provide valuable information, aiding in the prediction of flight delays, and the underlying features causing this. To get a better understanding and explore the data we are visualizing some of the data, enabling a deeper understanding of correlation between some of the features and the delay.

Data Visualization

To obtain a better understanding of our dataset, we have opted on using a range of visualizations in Tableau and Python to explore potential correlations between various features and the target value. To generate the visualizations effectively, considering the limitations that a big dataset poses, a stratified sample of the dataset was used of 100,000 rows.

The **LIGHT BLUE** indicates **NOT DELAYED FLIGHTS**
The **DARK BLUE** indicates **DELAYED FLIGHTS**

Our first objective was to depict the distribution of our dataset illustrating the proportions of our dataset corresponding to flights being delayed and not delayed (Figure 2). To do that we have used a pie chart, allowing us to present the proportions accurately and effectively. As a result, one can see that a larger portion of our dataset corresponds to flights not being delayed, yet a substantial part still represents flights being delayed. We have decided on using a pie chart as it offers a concise and quick overview of proportions among a restricted set of categories.

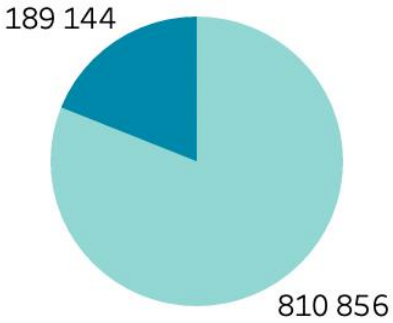


Figure 2 Distribution of target value

To identify areas where delays occur more often, we visualized this information on a geographical map as seen in figure 3 allowing location-based analysis thereby identifying hotspots. The different shades of blue effectively highlights the variations in the data, with darker shades indicating a higher average delay. One can conclude from the graph that there is a higher average delay to the East of America, in close proximity to the coast. This can be correlated with that some of America's biggest airports are located here.

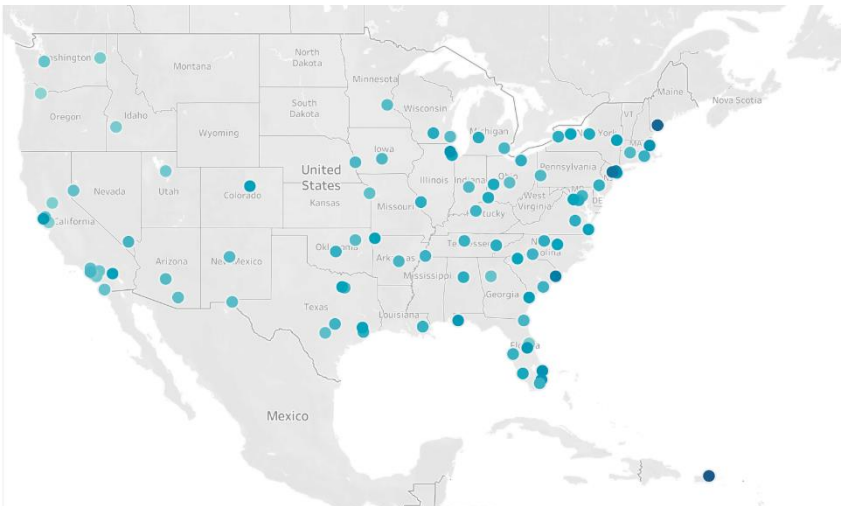


Figure 3 Average Delay across USA

Moreover, to investigate whether there are any correlations present within the columns in the dataset, a correlation matrix was created as depicted in figure 4. An appropriate color encoding scheme was used to represent the strength of correlation.

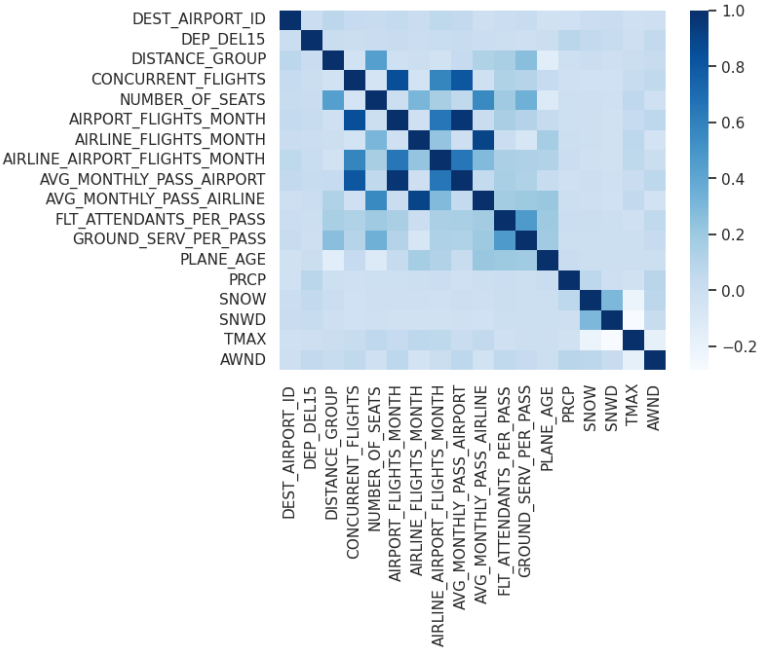


Figure 4 Correlation Matrix

To illustrate the distribution of delays throughout the day figure 5 was created which included the number of flights with a specific time block as well as the ratio of flights delayed and not delayed. By comparing occurrences between categories, a stacked bar chart in combination with colors can provide a useful overview. One can confidently conclude from the graph that delays are of a higher occurrence in the evening, though there are more flights at that time as well.

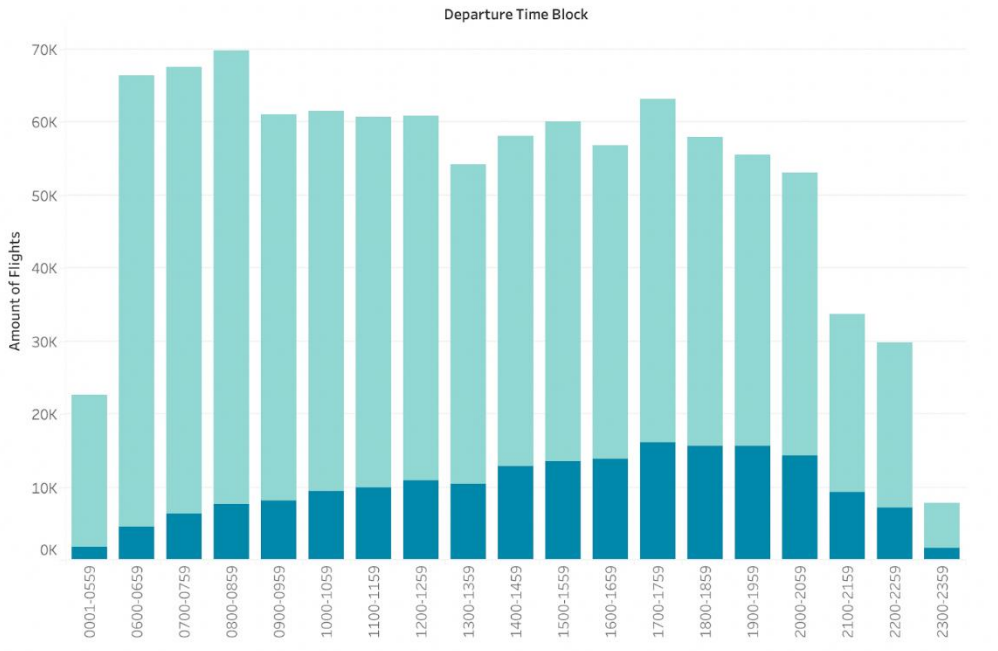


Figure 5 Distribution of delays during day

In addition to this, to investigate the influence of various weather conditions such as precipitation, snow, and inches of snow on the ground a side-by-side bar chart was created (Figure 6). The bars represent the incidence of the various weather conditions, categorized based on whether flights were delayed or not. Notably, a distinct correlation arises, where a higher occurrence of rain or snow corresponds to an increased probability of flight delays.

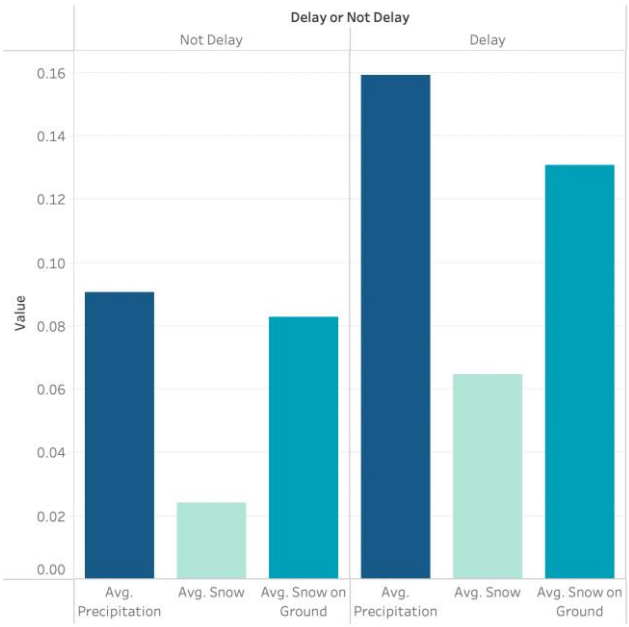


Figure 6 Weather vs. Target

The comprehensive range of visualizations generated was with a view to effectively present intricate data, thereby elevating the significance of this research. Every visualization created was substantiated by the various visualization principles. The principle of consistency enabled us to establish a cohesive visual identity, fostering a seamless viewing experience. The blue-teal color palette chosen was one that was more toned down, yet differences can be seen aiding in the differentiating of data. Using less saturated colors is visually more pleasing and less straining on the eyes. Maintaining consistency with the color schemes used, provided a more coherent look to the visualizations. Moreover, the visualization was kept simple and avoided unnecessary complexity, maximizing data to ink ratio. In addition, the structure of the visualization was done in a narrative manner, by first overviewing the dataset and then going into specific detail, depending on the correlations found, thereby guiding the readers through a coherent story within the data. Overall, using these visualization principles enabled us to significantly improve the visual quality of the research making it more engaging, informative, and visually appealing to readers.

Data Preprocessing

As mentioned earlier in the data description we are using a subset of the data to perform machine learning. To minimize computing time, we are using a stratified sample of 100,000 rows, which we consider enough for training and testing purposes and exceeds the rule of thumb that rows should be 10 times the columns from the original dataset (Smolic, 2022). The sample is stratified to ensure we keep the same proportions as in the full dataset. The dataset does not contain any NaN values that needs to be taken into consideration.

A selection of the features is objects or text strings. Not all models are able to interpret this data type and therefore they have been encoded as numerical values. Preferably all these objects would have been encoded using One-Hot Encoding to avoid creating a fictional hierarchy (ordinality) in the columns that do not exist (sarangpratap, 2022). However, some of the features contain many unique values that would cause too many columns (more than 6,000) and thereby making it almost impossible to compute. To validate this, we performed a test on a tuned KNeighborsClassifier model, and it produced almost identical results with full One-Hot Encoding and a split between label encoding and One-Hot Encoding. This type of encoding using both types simultaneously will later in data modeling be denoted as Split Encoding. In Split Encoding the columns DEP_TIME_BLK and ARR_TIME_BLK have been encoded with One-Hot-Encoding and the other object columns have been encoded with simple label encoding. This denotes the trade-off between the optimal solution and computational resources.

In some of the models full One-Hot Encoding will be used, when transparency is needed into the underlying factors (feature coefficients and trees need to be generated). This is to avoid ordinality in features that are nominal by nature. Thus, it does not seem to have a real impact on the results (the underlying factors in the prediction algorithm); a tree can, as an example, potentially introduce nodes that are based on this false ordinality, which will lower readability, transparency, and understanding.

The data sample has been split into a stratified train and test dataset with an 80/20 ratio. Since we will use cross-validation we don't need a validation dataset. The data will be trained and tested on different folds of the training data, and the best performing models will finally be tested on the test data without any further hyper parameter optimization to avoid potential overfitting to the testing data.

Data Modelling and Results

Our machine learning will be based on a wide range of different techniques and estimators that will be tuned with a selected range of hyperparameters each to ensure the best possible scores. These estimators include: KNeighborsClassifier, DecisionTreeClassifier, Logistic Regression, MLPClassifier, and RandomForestClassifier.

We will be using standardized data, using StandardScaler, for our gradient descent based and distance-based algorithms. This will ensure more leveled step sizes and weightage across the features (Chong, 2020). To avoid potential data leakage, we will be transforming the test data with the same estimator that was fitted with the training data (Brownlee, 2020).

Since our data is heavily imbalanced (minority only being around 19% of the sample) as stated earlier we will be using undersampling and oversampling techniques to try obtaining better results. Undersampling will be done by reducing the majority class (Not Delayed) using RandomUnderSampler by Imbalanced Learn, so that the classes are more evenly distributed.

We are also going to use SMOTE to oversample our minority class in another test, where synthetic samples are created for the minority. This will emphasize the features that exist in the delayed flights, since they are synthetically generated based on k_neighbors until an even distribution is made.

Both the sampling techniques will hopefully push the models towards finding correlations and disincentivize guessing on the majority class.

The following needs to be taken into consideration, when reading the modelling section. A delayed flight (Class 1) will be considered as a positive instance in this analysis. Random_state will be used whenever possible to ensure transparency and give the ability to later regenerate the results of the machine learning tasks done in this assignment. Our target is to be able to predict when a flight is going to be delayed, but also to understand which features are most likely to correlate with a delay.

All the scores and results can be seen in Figure 7 in the section Model Conclusion. In appendix 4, there are also a range of confusion matrixes available for further exploration, and appendix 3 includes a piece of the code that is used to tuning and generating results.

DummyClassifier

Due to the highly imbalanced nature of the data we need to take into consideration that a quite high accuracy can be obtained by doing educated guessing. To get an accuracy benchmark a DummyClassifier has been created (Müller & Guido, 2016, pp. 277-279) that just by guessing on the most frequent class (Not Delayed) got a score of 0.811 on the test data.

Since what we are really interested in is knowing how many true positives we can generate with the lowest error rate, F1-score will be used as our primary scoring method and the evaluation metric, when we are tuning the hyperparameters, because it combines the precision and recall in an attempt to summarize and score the best trade-off between the two. Accuracy, precision, and recall will also be taken into consideration, when looking at pros and cons of the models.

F1 Baseline

A good baseline to see if our model is finding any useful patterns in the data is to create a DummyClassifier that always predicts true (Zach, 2021). This will cause an incredibly low accuracy, but we will have a recall of 1, since there are no false negatives, but we will gather a number of false positives corresponding to the accuracy we just determined in our most frequent classifier. In this case the F1-score is 0.3181.

KNeighborsClassifier

This model is quite simple and transparent, and since we are only storing the training data and not doing any actual fitting, we normally also get quite good performance (Müller & Guido, 2016, pp. 20-21). Since this model is based on proximity or distance between points, they are especially vulnerable to using unscaled data. It is likely to be biased against bigger distances. Therefore, we are

also trying scaled versions of the data to test with. Since we cannot extract any coefficients or importance of features, we will be using the split encoded data.

We are getting quite high accuracy as expected on an untuned unscaled version, but the model still seems to be underfitting, because it does not seem to have found any useful patterns and performs worse on the testing data than the DummyClassifier.

With a StandardScaler the model has been approved to improve our model. We are getting a higher precision, thereby identifying more true positives only causing a few extra false positives. Since the proportional increase in the recall is bigger than in accuracy it seems like we are moving false negatives towards the true positives. Since precision and recall has increased, it also means that our F1 score has improved.

Tuned KNeighborsClassifier

To obtain even better results, we will be doing hyperparameter tuning on the estimator. This will be done using GridSearchCV, which takes a dictionary of parameters and exhausts all different combinations, while using cross-validation to test the results. Then it gives back the model that performs best from a set criterion. We will be using F1 as the scoring metric to select the best combination.

We will be using the scaled version of the data. The hyperparameters that will be tuned include: n_neighbors, weights, metrics.

The GridSearchCV returns that the parameters that give us the highest F1-score is the hamming metric, 1 neighbor, and a uniform weight. This gives a F1 value of 0.2689, which is an improvement from before. The model is overfitting on the data meaning the model is too complex since it is performing perfectly on the training data (1.0) and performing much worse on the testing accuracy (0.728). The model still gives a lot of false positives and false negatives meaning that the estimator is not good at predicting if the flight will be delayed, which is also indicated by that recall and precision are both below 0.5, which can be seen from the confusion matrix.

DecisionTreeClassifier

This model can provide useful insights since it builds a range of if/else questions that in the end will lead to a conclusion (Müller & Guido, 2016, pp. 70-71). We can extract the importance that is put into each feature and at the same time generate a visualization of the steps that are taken as part of the predictions. Scaled data has no effect on tree models, but we will be using the full One-Hot Encoded data to ensure that we don’t introduce false ordinality.

As a benchmark we fit the tree without any tuning to see if the tuning will cause a performance increase. The model is overfitting and thereby too complex, which is natural with a tree estimator like this since it will continue to ask questions until it is certain that it predicts the right class on the training data (pure leaf nodes). It is performing on a comparable level with the tuned version of the KNeighborsClassifier.

Tuned DecisionTreeClassifier

With GridSearchCV we have attempted to control the complexity by trying out different max_depth's and criterions. The parameters that gave the highest F1-score was the entropy criterion, but with full max_depth. Our model is almost performing the same, and is still overfitting on the data, but the recall has improved, sacrificing a small amount of precision and test accuracy. Some of the similarity is found in the fact that the only parameters that vary from the standard settings are the criterion. We are still using the full tree.

Feature Importance and Decisions

Our DecisionTreeClassifier can help provide insights into the features that are used in the decision making and the weight that is assigned to each of them. The feature importance ranges from a scale from 0 to 1, where 1 denotes perfect prediction (Müller & Guido, 2016, pp. 78-79).

It seems like the model is attempting to use almost all features, which makes the feature importance quite low. This is also indicated by the fact that there was almost no statistical correlation between any of the features and the target, resulting in the algorithm needing to utilize a combination of multiple features to be able to make correct predictions. The most important features are wind, maximum temperature of day, destination airport, concurrent flights, and the average number of flights for this airline in this particular airport (see appendix n).

The feature importance doesn't give us any information about if the features will be pushing towards a prediction of the class being delayed or not delayed.

The top of our decision tree can help provide insights into the first steps in the tree's decision making. Here it is seen that if the flight has been to another airport before, the rain level and if departure time is placed between 2200-2259 are the first steps that would be taken if the tree was only allowed two steps to make a decision. From this example tree we can conclude that if the flight was not at a previous airport and the departure time block was 2200-2259 the flight will be classified as delayed (Appendix 1).

RandomForestClassifier

To create a more powerful model we can use ensembles of models (Müller & Guido, 2016, p. 83). We have decided to use a RandomForestClassifier that combines multiple differentiated decision trees to improve predictions. Here we will be using full One-Hot Encoding, because we want to see the importance of the features. A not tuned version of this estimator gives a low F1-score. We only got a score of 0.0873. The precision is much better than in the previous estimators and we are not providing as many false positives as before.

Tuned RandomForestClassifier

We are tuning the criterion, max_depth, and number of estimators with GridSearchCV to control the complexity and try to improve the scores. Criterion should be gini, we should have all the nodes, and the number of estimators should be 1 to maximize the F1. This makes a model very similar to our single DecisionTreeClassifier. The model is being simplified and thereby reducing some of the overfitting, but we are in general performing worse than the single DecisionTreeClassifier with lower accuracy, precision, recall, and thereby F1-score.

Logistic Regression

Linear models are also transparent allowing us access to the coefficients that are pointing towards either class. Though being simple, it can sometimes provide very useful predictions. We are using the full One-Hot Encoding, because we want to analyze the features. To avoid the coefficients being affected by different scales of the features, we are using the StandardScaler as part of the pipeline.

The baseline model without tuning is almost as good on the testing data as on the training data, but the model is underfitting and lacking complexity, since we are only getting an accuracy around 0.8. The F1 score is not very good, because of a very low recall.

Tuned Logistic Regression

We have performed hyperparameter tuning on the LogisticRegression estimator. Originally, we did hyperparameter tuning on the solver and the C-value, but other solvers than 'lbfgs' took an exceptionally long time to compute and did not give a performance increase. Therefore, we have tuned using C-value to control the regularization of estimation. We ended up with a quite high value of 100 meaning that we have a low regularization and are focusing more on individual points. We almost got the same results as the not tuned version.

Feature Coefficients

The coefficients provide insight into which features are most important in determining either class. Features that are leaning towards a delayed flight are the two individual carriers SkyWest Airlines Inc. and Atlantic Southeast Airlines, the number of flights at the airport, the number of flights by the airline and the amount of rain.

It seems though airline carriers like Comair Inc. and American Eagle Airlines Inc. are not very often delayed and thereby pulling towards a not delayed classification. It does also seem that the departure time block 0600-0659 is causing few delays, and that having more flight attendants also helps minimize delays (Appendix 2).

Even though the model is not giving a very high prediction these coefficients help us to discover the features that actually provide an impact to the classification. The top and bottom features can give a foundation for further insight into how to minimize delays.

MLPClassifier

This MultiLayer Perceptron is the most advanced estimator used in this assignment. The combination of hidden layers, nodes, and weighted decision making creates a network that in some instances can provide strong predictions (Müller & Guido, 2016, pp. 104-106). This model is not transparent by nature and at the same time it is computationally expensive.

This model does not require, but benefits from scaled data to help outweigh distances. Therefore, it is being trained on the scaled split encoded version.

Accuracy is still underfitting meaning the model is too simple. The F1 score is competitive to some of the other estimators.

Undersampling and SMOTE

In an attempt to handle the imbalance that exists in the nature of the dataset, we have used two techniques as mentioned from the imbalanced-learn library. These include Synthetic Minority Oversampling Technique (SMOTE) and Random Undersampling. We will be trying these methods on our tuned DecisionTreeClassifier since it had the highest not dummy generated F1-score, and on the not tuned RandomForestClassifier, which had the highest precision.

SMOTE creates synthetic instances of the minority class (delayed flights), so we end up with an equal amount of both classes (*SMOTE*, n.d.). The SMOTE on the DecisionTreeClassifier did perform very similar to the original run, which could indicate that the decision tree did not obtain any new information and have not been very affected by the imbalance. On the RandomForestClassifier, SMOTE was able to give a higher F1-score, because of a significant increase in recall, and a proportionally less decrease in precision.

With the random undersampling technique we are not relying on synthetically generated instances, but instead we remove some of the majority class (not delayed) to ensure a balanced distribution (*RandomUnderSampler*, n.d.). Both on the RandomForestClassifier and DecisionTree-Classifier we had a decrease in precision, but the model was willing to classify as delayed causing correct predictions, which meant that the recall increased so much that the overall F1-score improved so much that it actually exceeded our benchmark.

Model Conclusion

The range of different machine learning estimators and techniques has allowed us to explore possible algorithms for predicting if a plane will get delayed. We are able to two estimators with undersampling over our F1-benchmark of 0.3181. As predicted, we did not obtain any significant accuracy increase with a mix of estimators either being too complex (overfitting) or being too simple/general (underfitting).

The most significant models that were obtained was the RandomForestClassifier and DecisionTreeClassifier with the RandomUnderSampler from the imblearn library. The model has though a low precision that would be causing to many false positives in a real life scenario.

Another discovery was the untuned RandomForestClassifier that as the only model successfully obtained a precision over 0.5 meaning that it provides more true positives than false positives. It is though still providing many false positives, which could cause unnecessary flags if the system was implemented.

This could indicate that there is no common reason for delays across all of the United States. Maybe small geographical areas, or other features can provide the basis needed to build a stronger algorithm in predicting the delays.

The LogisticRegression model and the DecisionTreeClassifier gave us valuable insight into some of the features that is causing delays. We discovered the impact of wind, rain, temperature, the business of an airport etc. These findings also support the discoveries that were made through the visualizations.

Estimator	Train Accuracy	Test Accuracy	Precision	Recall	F1	Tuned	Scaled	One-Hot Encoding
RandomForestClassifier - RandomUnderSampler	0.7580	0.6196	0.2816	0.6521	0.3934	False	False	True
DecisionTreeClassifier - RandomUnderSampler	0.7273	0.5557	0.2290	0.5699	0.3267	True	False	True
DummyClassifier - Constant Delayed	0.1891	0.1892	0.1892	1.0000	0.3181	False	False	False
DecisionTreeClassifier	1.0000	0.7310	0.2839	0.2773	0.2806	True	False	True
DecisionTreeClassifier - SMOTE	1.0000	0.7230	0.2708	0.2744	0.2726	True	False	True
KNeighborsClassifier	1.0000	0.7276	0.2730	0.2649	0.2689	True	True	False
DecisionTreeClassifier	1.0000	0.7452	0.2870	0.2337	0.2576	False	False	True
RandomForestClassifier	0.8985	0.7238	0.2436	0.2186	0.2304	True	False	True
MLPClassifier	0.8285	0.7906	0.3650	0.1443	0.2069	False	True	False
RandomForestClassifier - SMOTE	1.0000	0.8043	0.4348	0.1155	0.1825	False	False	True
KNeighborsClassifier	0.8346	0.7872	0.3300	0.1213	0.1774	False	True	False
KNeighborsClassifier	0.8281	0.7802	0.2568	0.0854	0.1281	False	False	False
KNeighborsClassifier	0.8281	0.7802	0.2568	0.0854	0.1281	False	False	False
LogisticRegression	0.8192	0.7965	0.3375	0.0788	0.1277	True	True	True
LogisticRegression	0.8192	0.7965	0.3375	0.0785	0.1274	False	True	True
RandomForestClassifier	1.0000	0.8118	0.5294	0.0476	0.0873	False	False	True
DummyClassifier - Most Frequent	0.8110	0.8110	0.0000	0.0000	0.0000	False	False	False

Figure 7 Model Results

Conclusion

The aviation industry causes a huge impact on society, and delays are very costly to the airline carriers and society. Machine Learning can potentially help enlighten the features that could indicate a delay and help predict delays in the industry improving customer satisfaction and profit. Through visualizations we were able to discover patterns in the data that would have almost impossible to identify just by looking at the data table. Several correlations were discovered during the exploratory

research. When planes were delayed there were on average more snow and rain. It is more likely that you will be delayed during the evening, where there are relatively many flights than during the night and day, and the bigger airports seem to have higher average delays. Especially the ones located on the east coast.

Through a range of estimators and techniques we were not able to provide the receipt for a machine learning algorithm that predicts if the flight was delayed, but interesting discoveries were made. Undersampling the data can provide a benefit to the F1-score balancing out the target classes without creating synthetic instances incentivizing that the model will predict on the minority class.

The models helped provide useful insights about the features that are leading to delays bringing a place where airline carriers and airports can start to look for improvements. For further research it is possible to discover these features more in depth. It will also be possible to try making predictions on a smaller geographical scale or with other features. This could potentially lead to algorithms and models that brings even greater insight into the features causing delays and predicting flight delays in advance.

References

2019 Airline Delays w/Weather and Airport Detail. (2022). [Dataset; Kaggle]. https://www.kaggle.com/datasets/threnjen/2019-airline-delays-and-cancellations

Asquith, J. (2020, April 6). If Aviation Was A Country It Would Be The World's 20th Largest By GDP. *Forbes*. https://www.forbes.com/sites/jamesasquith/2020/04/06/if-aviation-was-a-country-it-would-be-the-worlds-20th-largest-by-gdp/

Brownlee, J. (2020, August 17). How to Avoid Data Leakage When Performing Data Preparation. *MachineLearningMastery*. https://machinelearningmastery.com/data-preparation-without-data-leakage/

Chong, J. (2020, December 30). What is Feature Scaling & Why is it Important in Machine Learning? *Towards Data Science*. https://towardsdatascience.com/what-is-feature-scaling-why-is-it-important-in-machine-learning-2854ae877048

Dissanayaka, D. M. M. S., Adikariwattage, V., & Pasindu, H. R. (2019). *Evaluation of Emissions from Delayed Departure Flights at Bandaranaike International Airport (BIA)* (Vol. 186) [Atlantis Press]. https://doi.org/10.2991/apte-18.2019.26

Efthymiou, M., Njoya, E. T., Lo, P. L., Papatheodorou, A., & Randall, D. L. (2019). The Impact of Delays on Customers' Satisfaction: an Empirical Analysis of the British Airways On-Time Performance at Heathrow Airport. *Journal of Aerospace Technology and Management, 11*. https://doi.org/10.5028/jatm.v11.977

Khan, M. a. H., Uddin, M. F., Sarshaar, M. a. W., & Hashmi, J. (2022). Flight Delay Prediction Using Machine Learning. *Journal of Engineering Sciences, 13*(5), 254–261. https://jespublication.com/upload/2022-V13I5032.pdf

Müller, A. C., & Guido, S. (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists* (1st ed.). OReilly. http://cds.cern.ch/record/2229831

RandomUnderSampler. (n.d.). Imbalanced Learn. https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html

sarangpratap. (2022, June 22). Feature Encoding Techniques Machine Learning. *GeeksforGeeks*. https://www.geeksforgeeks.org/feature-encoding-techniques-machine-learning/

Smolic, H. (2022, December 15). How Much Data Is Needed For Machine Learning? *Graphite Note*. https://graphite-note.com/how-much-data-is-needed-for-machine-learning

SMOTE. (n.d.). Imbalanced Learn. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html#r001eabbe5dd7-1

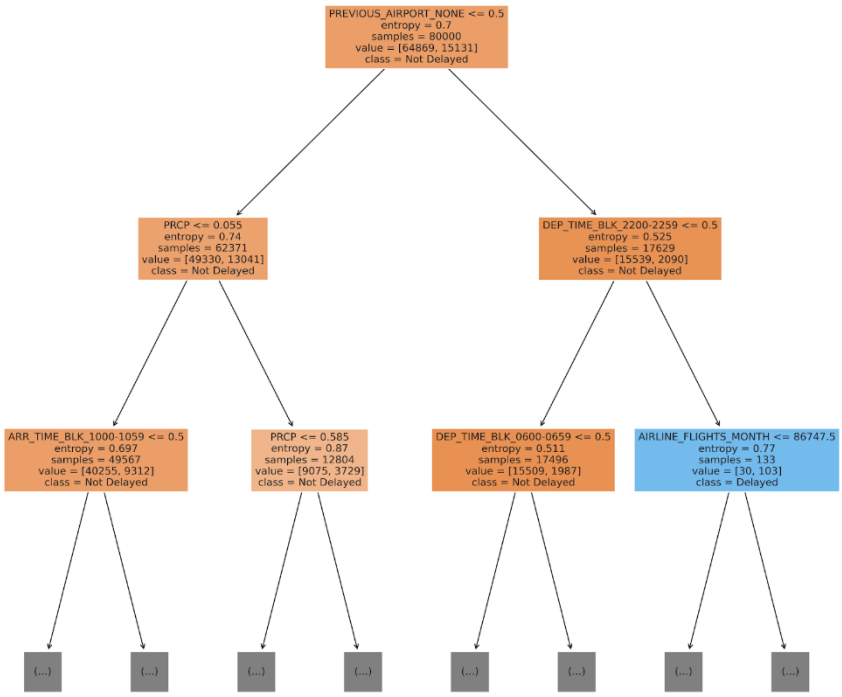
Types of Delay. (n.d.). Federal Aviation Administration. Retrieved May 23, 2023, from https://aspm.faa.gov/aspmhelp/index/Types_of_Delay.html

Vijayan, V. K. (2023, March 21). Flight Delays: What can be improved? *LinkedIn Pulse*. https://www.linkedin.com/pulse/flight-delays-what-can-improved-vinod-k-vijayan/

Zach. (2021, September 8). What is Considered a “Good” F1 Score? *Statology*. https://www.statology.org/what-is-a-good-f1-score/

Appendix 1. Feature Importance and tree from DecisionTreeClassifier

Features	Feature Importance
AWND	0.069016
TMAX	0.068040
DEST_AIRPORT_ID	0.062723
CONCURRENT_FLIGHTS	0.056959
AIRLINE_AIRPORT_FLIGHTS_MONTH	0.054148
PLANE_AGE	0.051108
AIRPORT_FLIGHTS_MONTH	0.039729
DAY_OF_WEEK	0.035605
PRCP	0.032706
AIRLINE_FLIGHTS_MONTH	0.032555
NUMBER_OF_SEATS	0.028292
DISTANCE_GROUP	0.026164
MONTH	0.024687
AVG_MONTHLY_PASS_AIRPORT	0.019591
PREVIOUS_AIRPORT_NONE	0.010299



Appendix 2. Coefficients from LogisticRegression

Features	Coefficients
CARRIER_NAME_SkyWest Airlines Inc.	0.477400
AIRPORT_FLIGHTS_MONTH	0.383424
CARRIER_NAME_Atlantic Southeast Airlines	0.226056
AIRLINE_FLIGHTS_MONTH	0.209623
PRCP	0.190826
PREVIOUS_AIRPORT_Ogdensburg International	0.115425
AWND	0.107515
DEP_TIME_BLK_1700-1759	0.091647
PREVIOUS_AIRPORT_NONE	0.090518
ARR_TIME_BLK_2300-2359	0.087438
TAIL_NUM_N837AE	0.087048
ARR_TIME_BLK_2100-2159	0.085994
DEPARTING_AIRPORT_Newark Liberty International	0.085138
TAIL_NUM_N8710M	0.085000
CARRIER_NAME_JetBlue Airways	0.083161

Features	Coefficients
CARRIER_NAME_Comair Inc.	-0.491876
CARRIER_NAME_American Eagle Airlines Inc.	-0.403183
DEP_TIME_BLK_0600-0659	-0.260649
FLT_ATTENDANTS_PER_PASS	-0.246566
PREVIOUS_AIRPORT_Ford	-0.203878
TAIL_NUM_N430WN	-0.198267
TAIL_NUM_N522LR	-0.186567
TAIL_NUM_N451WN	-0.179058
PREVIOUS_AIRPORT_Great Falls International	-0.178982
TAIL_NUM_N247SY	-0.178012
PREVIOUS_AIRPORT_Dothan Regional	-0.177071
TAIL_NUM_N8696E	-0.173494
TAIL_NUM_N909DE	-0.170993
TAIL_NUM_N12564	-0.170825
TAIL_NUM_N406YX	-0.169848

Appendix 3. Example Code of Tuning

```
# The scoring methods that needs to be calculated
scoring = {'accuracy': 'accuracy', 'precision': 'precision', 'recall': 'recall', 'f1': 'f1'}

# The different parameters that should be searched by GridSearchCV
param_grid = {
    'kneighborsclassifier__weights': ['uniform', 'distance'],
    'kneighborsclassifier__n_neighbors': list(range(1, 15, 2)),
    'kneighborsclassifier__metric': ['manhattan', 'euclidean', 'hamming', 'minkowski', 'chebyshev']
}

# Our pipeline that first scales the data without leakage and then fits the model
model = make_pipeline(StandardScaler(), KNeighborsClassifier())

# The grid is build. We are using f1 as our selection method
grid = GridSearchCV(model, param_grid, cv=3, refit='f1', scoring=scoring, return_train_score=True, verbose=3)

# We are fitting the training data
grid.fit(X_train_enc, y_train_enc)

# We are doing a prediction on our test data (used for confusion matrix)
y_pred = grid.predict(X_test_enc)

# Confusion matrix is generated based on the predictions and the actual labels
cm = confusion_matrix(y_test_enc, y_pred, labels=grid.classes_)

# Utilizing a customized function of previous work: https://github.com/DTrimarchi10/confusion_matrix
make_confusion_matrix(cm, group_names=labels, categories=categories, title='Tuned and Scaled', model='KNeighborsClassifier', model_type='neighbors', params=grid.best_params_)

# Generating Different Metrics
print(f'True Positives: {cm[1][1]}')
print(f'False Positives: {cm[0][1]}')
print(f'Training Accuracy: {grid.best_estimator_.score(X_train_enc, y_train_enc):.4f}')
print(f'Testing Accuracy: {grid.best_estimator_.score(X_test_enc, y_test_enc):.4f}')
print(f'Precision: {precision_score(y_test_enc, y_pred):.4f}')
print(f'Recall: {recall_score(y_test_enc, y_pred):.4f}')
print(f'F1: {f1_score(y_test_enc, y_pred):.4f}')
```

Appendix 4. A selection of confusion matrixes