Machine Learning Analysis of Flight Delays

Tianzi Qin
Yedian Cheng
Baozhang Min

1. Introduction

1.1 Background and Importance of the Study

Flight delays are a global issue that significantly impacts the aviation industry, passengers, and broader economic sectors. These delays can arise from many factors, including adverse weather conditions, technical issues, air traffic control restrictions, and more. For airlines, flight delays lead to increased operational costs, disruptions in scheduling, and potential loss in revenue. From a passenger perspective, these delays not only cause inconvenience and discomfort but can also lead to missed connections and notable events, resulting in a negative travel experience. The cumulative effect of these delays can ripple through the economy, affecting not just the aviation sector but also tourism, business operations, and the service industry.

In recent years, machine learning has emerged as a powerful tool to tackle complex and dynamic problems across various domains, and the issue of flight delays is no exception. Machine learning's ability to analyze large datasets and uncover hidden patterns makes it exceptionally suited for predictive analytics in the aviation field. By leveraging historical data on flight schedules, weather patterns, and airport traffic, machine learning algorithms can predict potential delays more accurately than traditional statistical methods. This predictive capability is crucial for proactive planning and decision-making, both for airline operators and passengers. The dynamic and multi-faceted nature of factors leading to flight delays necessitates more advanced analytical methods.

Traditional approaches often fall short in capturing the complexities and interdependencies of the variables involved. Machine learning, with its advanced algorithms and data-processing capabilities, can bridge this gap. It can not only predict delays with higher accuracy but also offer insights into the underlying causes and potential mitigation strategies. The application of machine learning in this context promises not just incremental improvements in delay management but a transformative approach to how flight operations are optimized and how passenger experience is enhanced.

1.2 Scope and Limitations of the Study

1.2.1 What We Intend to Investigate:

- Use of various data points (airline, flight number, airports, scheduled and actual times, distance, etc.) to predict flight delays. The specific time window for data points is before the flight takes off, and we want to predict whether the flight is delayed when arriving. This forms the core of our project, where we will actively apply and evaluate different predictive models.
- The theoretical impact of predictive analytics on reducing uncertainty and improving the passenger experience. We aim to discuss and theorize how our findings could be used to enhance passenger comfort and reduce the stress associated with air travel.
- The potential role of predictive modelling in improving airline operational efficiency. We will focus on exploring the possibilities and discussing how airlines might use our predictive models to streamline operations and manage resources more effectively, rather than on practical application.

1.2.2 What We Will Not Explore:

- In-depth technical details of flight mechanics and engineering.
- Primary research like surveys or interviews with airline staff or passengers.

1.3 Significance of the study

In our coursework, we have been introduced to core concepts of machine learning, including supervised versus unsupervised learning, clustering with the k-means algorithm, and regression techniques such as linear regression and the gradient descent algorithm. However, our understanding remains theoretical, and we have not applied these concepts in practical, real-world scenarios.

Through this project, we aim to bridge this gap. By applying machine learning techniques to predict flight delays, we not only deepen our understanding of these methods but also contribute to a practical solution for a prevalent industry challenge. This project will enable us to transition from theoretical knowledge to practical application, honing our skills in data analysis, model building, and problem-solving. Additionally, it offers the potential to provide valuable insights to airlines and passengers, aiding in better decision-making and improving the overall travel experience.

1.4 Personal Importance

- **Yedian Cheng:** I am deeply fascinated by the environmental impact of flight delays, notably the increased fuel consumption and resulting carbon emissions. My focus is on how predictive models can help make flight operations more eco-friendly. By accurately predicting delays, we can assist airlines in optimizing schedules and routes, potentially reducing fuel usage and emissions. My goal is to show that data analysis can be an effective tool for achieving more sustainable aviation practices.
- **Tianzi Qin:** Flight delay prediction is a challenging problem that requires analyzing complex datasets to find patterns and make accurate predictions. The delay is influenced by factors, including weather, traffic, technical issues, and human factors. Understanding and analyzing these complicated factors could sharpen my data science skills. Moreover, by predicting delays, we could contribute to enhancing the travel experience for millions of passengers, making it a socially impactful endeavor.
- **Baozhang Min:** I have frequently encountered flight delays and cancellations over the years. These delays often involved extended periods of 'waiting', a situation where passengers and staff were left without advance notice. Cancellations were sometimes even decided after the crew had exceeded their allowed work hours. This 'waiting' resulted in a significant waste of time and energy, bringing additional and unnecessary fuel emissions during extended engine idle times. Witnessing these challenges, myself, I recognize the remarkable benefits of improving prediction models to enhance the passenger experience, reduce costs, and minimize environmental impact.

2. Methodology

2.1 Data Collection

Our dataset is from Kaggle: https://www.kaggle.com/datasets/usdot/flight-delays/data.

It contains three datasets: airlines.csv, airports.csv, and flights.csv. The first two files are reference files for airline and airport codes. The third file is the core file containing flight delayed information and all variables that can be used to predict, all columns' definitions could be checked on the Kaggle website. The original dataset has more than 5.8 million rows, due to the limitation of computer memory, we chose 2 million rows to analyze data distribution and chose 500 thousand rows for model training.

The project utilizes a comprehensive dataset comprising flight details for the year 2015. This dataset was sourced from a publicly available repository, ensuring accessibility and transparency. It includes a variety of parameters such as airline details, flight numbers, departure and arrival times, delay durations, and more, offering a rich ground for analysis.

Columns definition in flights.csv:
https://www.kaggle.com/datasets/usdot/flight-delays/data?select=flights.csv

2.2 Data Preprocessing and techniques:

Data was imported using Python's Pandas library, a powerful tool for data manipulation and analysis. Initial data preprocessing steps included handling mixed data types, managing missing values, and converting data types for consistency and accuracy.

2.2.1 Pandas

Pandas is a foundational tool in Python for data manipulation and analysis. In our project, it played a crucial role in importing, cleaning, and structuring the flight data. Key features of Pandas that were utilized include:

- DataFrames and Series: These are the primary data structures in Pandas. Data Frames allow for storing and manipulating tabular data in rows of observations and columns of variables.
- Data Cleaning: Functions such as dropna(), fillna(), and type conversions (like astype()) help in handling missing or incorrect data, a common issue in real-world datasets.
- Data Exploration: Pandas offers extensive capabilities for slicing, indexing, and filtering data, making it easier to segment the flight data based on specific criteria.
- Descriptive Statistics: With built-in functions like mean(), std(), and describe(), Pandas aids in quickly summarizing key statistical aspects of the data.

2.2.2 NumPy

NumPy is another integral Python library, particularly for numerical computing. In our project, NumPy would be instrumental in:

- Mathematical Operations: Handling mathematical calculations required for data analysis, such as computations on arrays or matrices.
- Efficient Storage: Offering an efficient way to store and manipulate large arrays of numerical data, which is critical when dealing with extensive datasets like flight records.
- Integration with Pandas: NumPy works seamlessly with Pandas, providing support for more complex mathematical operations that might be needed in the analysis.

2.2.3 Scikit-learn

Scikit-learn is a leading machine learning library in Python. It played a significant role in the predictive modeling aspect of our project:

- Preprocessing Data: Scikit-learn offers tools for preprocessing data, such as scaling features, encoding categorical variables, and splitting data into training and testing sets.
- Machine Learning Algorithms: It provides a wide array of supervised and unsupervised learning algorithms. In our case, algorithms like Logistic Regression, Decision Trees, Random Forest could be implemented using Scikit-Learn's standardized interface.
- Model Evaluation: Scikit-learn includes functions for cross-validation and various metrics to evaluate the performance of the models, essential for understanding and improving the predictive models.
- Pandas and NumPy form the backbone for data handling and numerical computations, while Scikit-learn provides a comprehensive suite of machine learning tools for predictive modeling. Together, they create a powerful combination for tackling complex data analysis tasks such as predicting flight delays and cancellations.

2.2.4 Matplotlib and Seaborn

Matplotlib and Seaborn are widely used Python libraries for creating static, animated, and interactive visualizations.

- Data Visualization: Allow for the creation of a wide range of graphs and plots, which are essential for visual data analysis. For instance, histograms, scatter plots, line charts, and bar graphs can be used to visualize various aspects of flight data, such as the frequency of delays, distribution of flight durations, or trends over time.
- Exploratory Data Analysis (EDA): Effective visualizations are crucial in EDA, helping to uncover underlying patterns, spot anomalies, and formulate hypotheses. With Matplotlib and Seaborn, we can graphically represent data distributions and relationships between variables, making it easier to understand complex datasets.

2.2.5 XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way.

2.2.6 LightGBM

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel, distributed, and GPU learning.
- Capable of handling large-scale data.

2.3 Algorithms

In this project, we did not develop new algorithms; instead, we strategically employed a range of established machine learning algorithms, each selected for its unique strengths and its ability to effectively tackle specific challenges presented by the flight data. This array of algorithms includes single model learning: Logistic Regression, Decision Trees and ensemble learning: Random Forests, XGBoost, and LigthGBM. Each algorithm has been chosen not only for its individual performance but also for how it complements the others in addressing the multifaceted nature of flight delay prediction.

2.3.1 Logistic Regression

Basic Concept:

Logistic Regression is a versatile statistical model primarily used for binary classification problems. In the realm of flight prediction, it can be adeptly employed to predict outcomes such as whether a flight will be delayed or not. The model works by estimating the probability of a binary response based on one or more predictor variables. It outputs probabilities between 0 (no chance of the event happening) and 1 (certain that the event will happen). For example, it can analyze historical flight data, considering factors like weather conditions, aircraft type, and airport traffic, to calculate the likelihood of flight delays. Despite its simplicity, Logistic Regression can be remarkably effective, especially when relationships between variables are approximately linear and when the dataset is not too large or complex.

Theories:

In contrast to linear regression where the dependent variable y is continuous, logistic regression where the dependent variable is a binary value of 0/1 requires us to create a mapping that converts the original real value to a 0/1 value. This requires us to create a mapping that converts the original real value to a 0/1 value. That's why we use sigmoid function:

$$h(x)= \frac{1}{(1+e^{-x})}$$

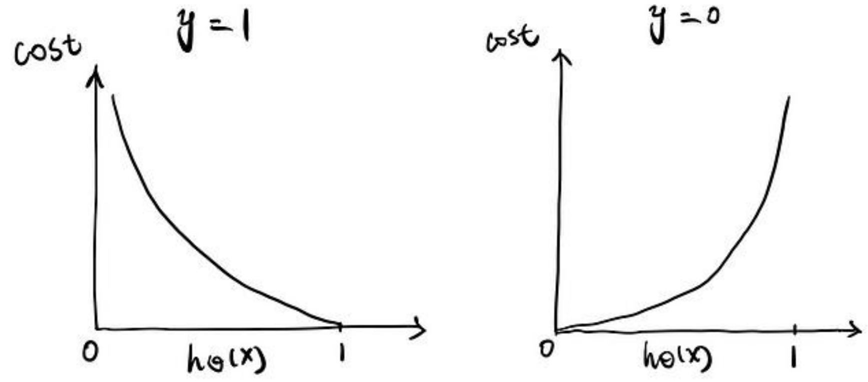
When h(x)>=0.5, y=1, h(x)<0.5, y=0

The loss function can be divided into two categories:

If the true category of the given sample y=1, the smaller the estimated probability p is, the larger the loss function is (estimation error)

If the true category of the given sample y = 0, the larger the estimated probability p, the larger the loss function (estimation error)

If y=1, $Cost\big(h\theta(x),\ y\big) =- \log\big(h\theta(x)\big)$
If y=0, $Cost\big(h\theta(x),\ y\big) =- \log\big(1 - h\theta(x)\big)$



it can be written as one single formula which brings convenience for calculation:

$$Cost\big(h\theta(x),\ y\big) =- y\log\big(h\theta(x)\big) - (1-y)\log\big(h\theta(x)\big)$$

The cost function for all data samples is:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

$$J(\theta) = \frac{1}{m}[\sum_{i=1}^{m} -y^{(i)} log(h_\theta(x^{(i)})) + (1-y^{(i)}) log(1-h_\theta(x^{(i)}))]$$

$$m = number\ of\ samples$$

Which can use gradient descent to find parameters that minimize the cost function.

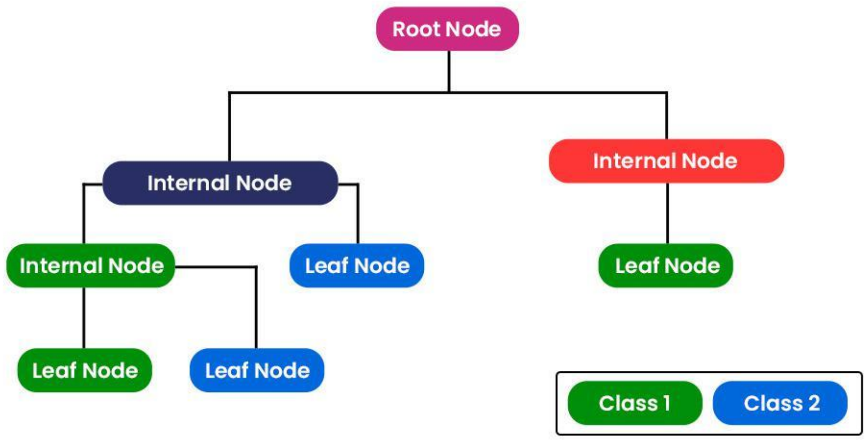
2.3.2. Decision Tree

Basic Concept:
The Decision Tree model is intuitive and widely used for classification and regression tasks. Its straightforward structure, resembling a tree, makes it an excellent tool for flight delay predictions. Each internal node of the tree represents a decision based on certain attributes of the data (like the time of day, weather conditions, or airline), and each branch represents the outcome of that decision, leading to a leaf node that signifies a prediction outcome. For instance, a decision tree might split the data first on weather conditions, then on the time of day, and so on, until it makes a final prediction about a flight's punctuality. While easy to interpret and implement, decision trees can suffer from overfitting, especially if they grow complex by fitting too closely to the training data.

Theories:
In our model we used the CART algorithm for the decision tree. A complete CART algorithm consists of three parts: feature selection, decision tree generation, and decision tree pruning. The CART algorithm consists of two main types: regression trees and classification trees. The regression tree is used for modeling tasks where the target variable is continuous, and the criterion for feature selection is minimizing the squared error. The classification tree is used for

modeling tasks where the target variable is discrete, and the feature selection criterion is the Gini Index.
The CART algorithm works via the following process:

- The best-split point of each input is obtained.
- Based on the best-split points of each input in Step 1, the new "best" split point is identified.
- Split the chosen input according to the "best" split point.
- Continue splitting until a stopping rule is satisfied or no further desirable splitting is available.



Gini Index:

$$Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2$$

To the given set D, K is the number of classes, C_k is the number of samples belonging to the kth class.
Gini (D, A):

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

Set D According to the feature A Whether it takes a certain possible value a is cut into D1 and D2, then feature A Conditional on the fact that the set of the Gini index is Gini (D, A).
Gini index represents the uncertainty of the set, the higher the Gini is, the higher the uncertainty is.
Calculate the Gini index for each feature of the dataset. For each feature A, for each of its possible value a, slice the dataset into two parts and calculate the Gini index. Select the feature with the smallest Gini coefficient and its cut point as the optimal feature and optimal cut point. A Gini Index of 0 indicates that all items in the node belong to a single class, implying perfect purity. As the variability in class increases, the Gini Index moves towards 1, indicating impurity. Keep looping until the condition is satisfied and stop.

2.3.3. Random Forest

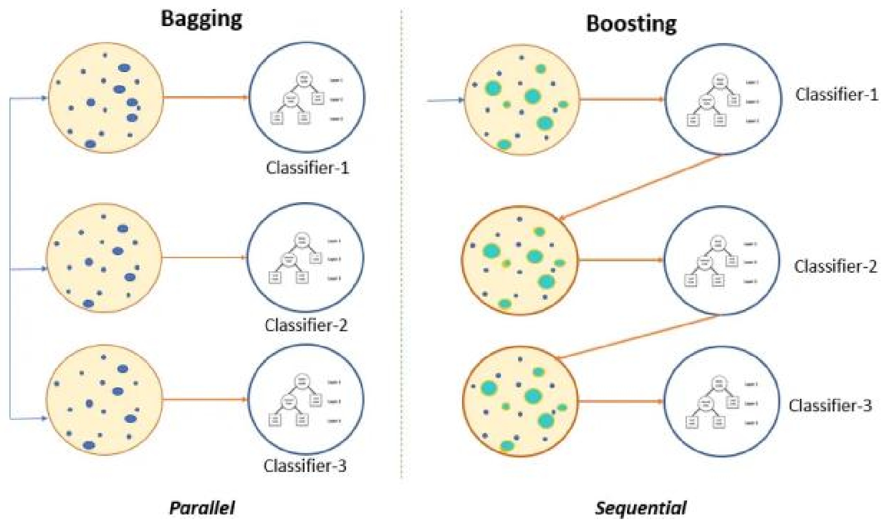
Basic Concept:

Random Forest is an ensemble learning technique that builds upon the simplicity of decision trees but enhances prediction accuracy and robustness. It creates a 'forest' of numerous decision trees, each trained on random subsets of the dataset, and aggregates their predictions to make a final decision. This method is highly effective in flight-delay prediction as it combines the insights of multiple trees, reducing the risk of overfitting and increasing the model's generalization capabilities. It can handle diverse and high-dimensional data, making it well-suited for complex problems like predicting flight delays, where numerous factors are at play. The model's ability to provide important scores for each feature also helps in understanding which factors most significantly impact flight delays.

Theories:

Bagging is the most typical representative framework for parallel integrative learning methods. The core concept is bootstrap sampling. Given a dataset containing m samples, a sample is randomly selected and put into the sampling set, and after m samples, a sample set of the same size as the original dataset is obtained. We sample T sample sets containing m samples, and then train a base learner based on each sample set and finally combine these base learners. This is the main idea of Bagging.

The diagram of Bagging and Boosting is shown below:



The process of random forest:

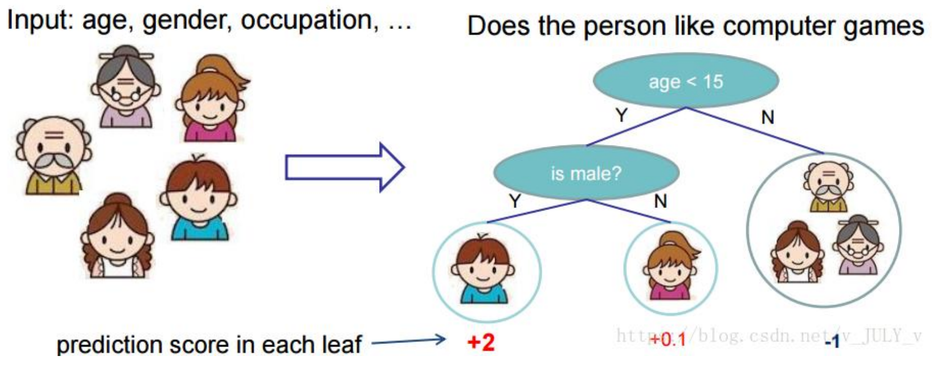
- Assuming there are M samples, we perform sampling with replacement to select M samples (each time randomly selecting one sample and then putting it back before the next selection).
- Suppose there are N features in the dataset. During the decision-making at each node of a tree, when a split is required, we randomly select n features from these N features, where n is much smaller than N. We choose the best feature from these n features to split the node. Based on the sampled M samples and n features, we construct a decision tree following this node splitting approach.

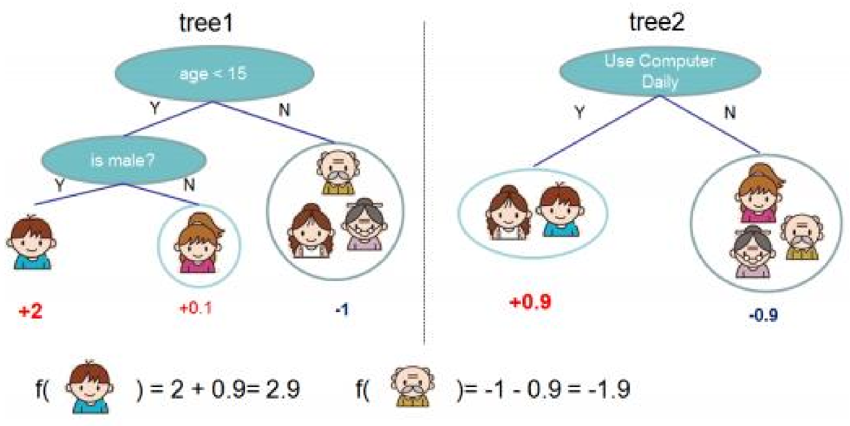
- We build a large number of such decision trees to form a Random Forest. The results of each tree are then aggregated to make the final prediction (using voting for classification tasks or averaging for regression tasks).

2.3.4. XGBoost

Basic Concept:
XGBoost stands out as a highly efficient and versatile implementation of gradient boosting machines. It is designed for speed and performance, making it a top choice for complex predictive modeling tasks like flight delay prediction. XGBoost builds multiple decision trees sequentially, where each new tree corrects the errors made by the previous ones. This method results in a powerful and accurate model capable of handling large datasets with a mix of categorical and numerical features. In the context of flight predictions, XGBoost can efficiently process and learn from a myriad of factors, from weather conditions and air traffic to more subtle indicators, to predict delays or cancellations with high accuracy. Its ability to handle overfitting, provide feature importance, and work with missing data makes it a robust choice for predictive analytics in aviation.

Theories:
The core theory of XGBoost is Gradient Boosting Decision Trees (GBDT), In GBDT, each new tree is built on the errors or shortcomings of the existing ensemble of trees. Gradient Descent is used to minimize a loss function. GBDT applies gradient descent to the problem of finding the best model. It does this by sequentially adding new trees that address and reduce the errors made by the previous trees.
Given an example, we want to predict the preference level for video games among family members, considering that younger people are more likely to enjoy video games compared to older people, and males tend to like video games more than females. The approach involves initially dividing the family into children and adults based on age, and then further distinguishing them by gender. Everyone is scored on their preference for video games. Two decision trees, tree1 and tree2, were trained following the principle of Gradient Boosting Decision Trees (GBDT), where the final prediction is the sum of the conclusions from both trees. Therefore, the predicted score for a child is the sum of the scores they receive in the corresponding nodes of both trees: 2 + 0.9 = 2.9. Similarly, the predicted score for a grandfather would be: -1 + (-0.9) = -1.9.

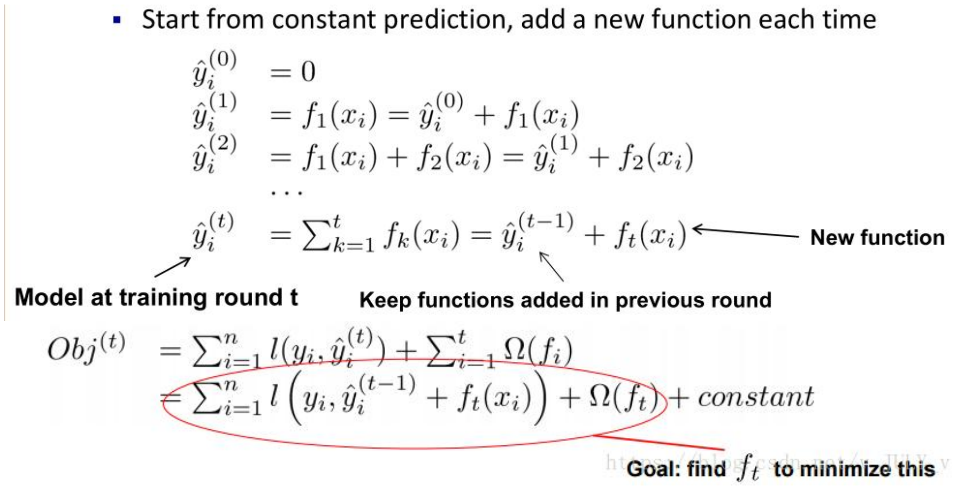

Input: age, gender, occupation, ...
Does the person like computer games
age < 15
Y
N
is male?
Y
N
prediction score in each leaf
+2
+0.1
-1
http://blog.csdn.net/v_JULY_v



The process of XGBoost is:

1. Trees are continuously added, and features are split to grow a tree. Each time a new tree is added, it essentially learns a new function f(x) to fit the residuals of the last prediction.
2. Once we have trained k trees and need to predict a score for a sample, it's done based on the features of the sample. In each tree, the sample will fall into a specific leaf node, and each leaf node corresponds to a score.
3. Finally, the predicted value for the sample is obtained by summing up the scores from the corresponding leaf nodes of each tree.

Loss function:



- Start from constant prediction, add a new function each time

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\cdots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

New function
Model at training round t
Keep functions added in previous round

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant
\end{aligned}
$$

Goal: find f_t to minimize this

2.3.5 LightGBM

Basic Concept:

LightGBM, short for Light Gradient Boosting Machine, is an advanced implementation of gradient boosting framework. The core components of LightGBM are decision trees, specifically gradient-boosted decision trees. Unlike other boosting frameworks that build trees depth-wise or level-wise, LightGBM grows tree's leaf-wise (best-first). It chooses the leaf with the highest delta loss to grow, leading to faster convergence and better accuracy with lower memory usage. LightGBM can also handle categorial features like airline names and airport codes efficiently, which is particularly suitable for features with high cardinality. It is also faster in training compared to other gradient boosting frameworks.

Theories:
LightGBM is also based on GBDT model but improves the shortcomings of XGBoost. XGBoost uses a pre-sorted algorithm to find the optimal splitting points of features. While the pre-sorted algorithm can accurately determine the split points, it consumes a lot of memory space. This can significantly affect the performance of the algorithm when dealing with datasets that have a large number of observations and features. To reduce the number of feature split points and more efficiently find the best splitting points, LightGBM, differing from XGBoost's pre-sorted algorithm, adopts a histogram-based algorithm. The basic idea is to discretize continuous floating-point feature values into k integers and construct a histogram with a width of k. When traversing the data of a particular feature, the discretized value is used as an index for cumulative statistics in the histogram. After one traversal, the histogram accumulates the corresponding statistical measures, and then the best split point is found based on this histogram.

2.4 classification model evaluation metrics

In machine learning, classification tasks are about predicting categorical labels for given input data. A pertinent example in the airline industry is the prediction of flight delays. This involves classifying flights into categories such as 'delayed' or 'on-time' based on various factors like weather conditions, mechanical issues, air traffic, etc. The effectiveness of such a classification model is pivotal in managing airline schedules, passenger convenience, and operational efficiency.
Evaluation metrics are crucial for quantitatively assessing the performance of flight delay prediction models. They help in gauging how accurately and reliably a model can predict delays, which is essential for operational planning and passenger communication. Different metrics are suited for different aspects of flight delay prediction, especially considering the imbalance that might exist between the number of on-time and delayed flights.

2.4.1 Accuracy

$$\boldsymbol{Accuracy = \frac{TP + TN}{TP + TN + FP + FN}}$$

In the context of predicting flight delays, accuracy is a straightforward measure indicating the proportion of correct predictions made by the model (both delayed and on-time flights). High accuracy is desirable, as it reflects the model's overall reliability. However, this metric can be misleading if most flights are on time; a model could show high accuracy simply by predicting no delays. Therefore, while accuracy is a good starting point for evaluating the model's performance, it doesn't fully capture the intricacies of flight delay predictions, especially in cases where delays are less frequent but critical to predict.

2.4.2 Precision

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Precision is particularly significant in the airline industry, where over-predicting delays can be as problematic as missing them. A model with high precision accurately identifies flights that will be delayed, minimizing false alarms. This is crucial for efficient resource allocation, such as crew scheduling and gate assignments. For passengers, precise predictions mean reliable information for planning their travel. However, focusing solely on precision might lead to a model that is overly cautious, predicting delays only in clear-cut cases and missing subtler patterns that could indicate potential delays.

2.4.3 Precision Recall (Sensitivity)

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Recall, or sensitivity, is a critical metric in scenarios where missing an actual delay is costly. A high recall rate means the model successfully identifies a large proportion of actual delays. This is essential for operational planning in the airline industry. It allows for proactive measures like rebooking passengers, adjusting crew schedules, and managing airport logistics. In the context of customer satisfaction, a high recall rate ensures that passengers are timely informed about potential delays, improving their travel experience. However, a model with high recall might also predict more false positives (non-delayed flights predicted as delayed), leading to inefficiencies.

2.4.4 F1 Score

The F1 score is particularly relevant when seeking a balance between not over-predicting and not under-predicting flight delays.

$$F1 = 2.\ \frac{Precision\ \times Recall}{Precision + Recall}$$

This metric is the harmonic mean of precision and recall, providing a single score that balances the two. A high F1 score indicates a model that is both accurate in its predictions of delays (precision) and comprehensive in catching most of the delays that occur (recall). In the airline industry, this balance is key to optimizing both operational efficiency and customer satisfaction. It ensures that the model is neither too conservative nor too liberal in predicting delays.

2.4.5 ROC Curve and AUC

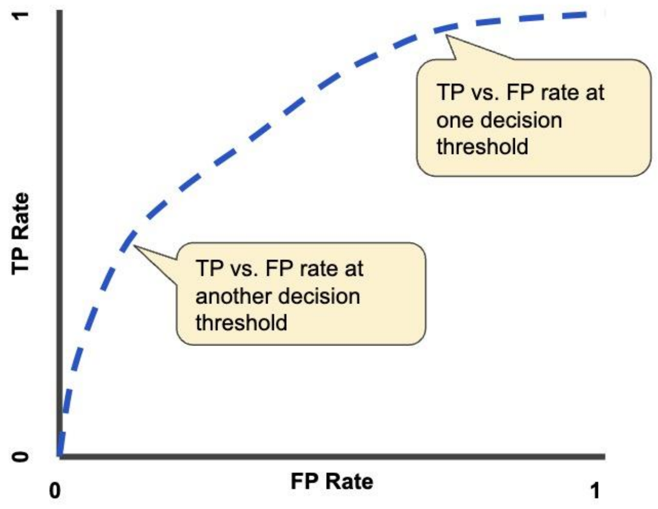
- ROC Curve (Receiver Operating Characteristic Curve)

  Fundamental Concept: The ROC curve is a graphical representation that illustrates the diagnostic ability of a binary classifier, such as a flight delay prediction model. It plots two parameters:
  True Positive Rate (TPR): Also known as sensitivity, recall, or hit rate, TPR is the ratio of correctly predicted positive observations (delayed flights) to the total actual positives (all delayed flights).

$$TPR = \frac{TP}{TP + FN}$$

  False Positive Rate (FPR): It represents the ratio of incorrectly predicted positive observations (flights wrongly predicted as delayed) to the total actual negatives (all on-time flights). FPR is calculated as FPR = FP / (FP + TN).

$$FPR = \frac{FP}{FP + TN}$$

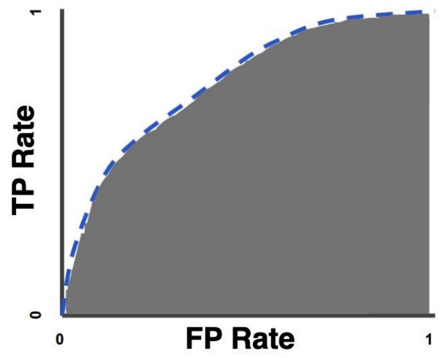


  In Flight Delay Prediction: The ROC curve helps airlines understand how well their model can distinguish between two classes: delayed and on-time flights. The curve is created by plotting TPR against FPR at various threshold settings. By analyzing this curve, airlines can assess the trade-offs between correctly identifying delayed flights and mistakenly classifying on-time flights as delayed.

- AUC (Area Under the ROC Curve, also referred to as ROC Score)

  Definition and Significance: AUC stands for Area Under the Curve. It measures the entire two-dimensional area underneath the ROC curve. AUC provides a single number summary of the model's performance by aggregating the performance at all possible thresholds.

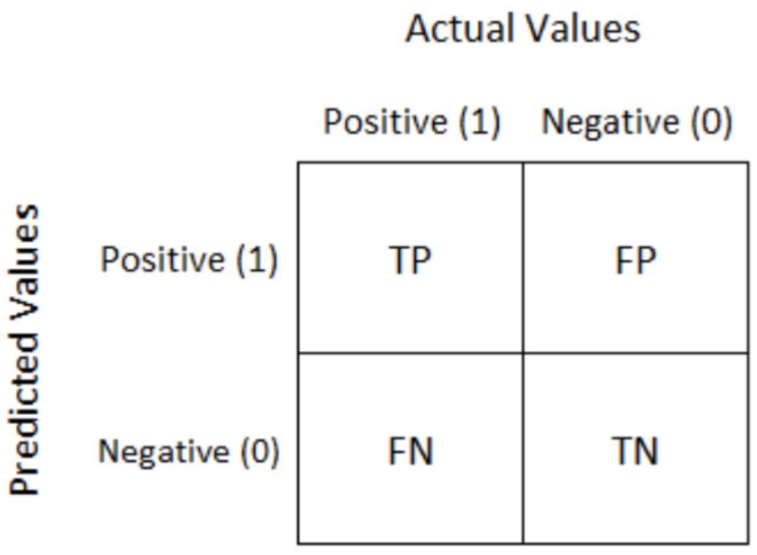
Interpreting AUC Values: An AUC of 1 represents a perfect model that perfectly distinguishes between delayed and on-time flights. An AUC closer to 0.5 suggests a model with no discriminative ability, equivalent to random guessing.



Application in Flight Delay Models: A high AUC value indicates that the model has a good measure of separability, meaning it can effectively distinguish between delayed and on-time flights. For airlines, a model with a higher AUC is more reliable and efficient for predicting flight delays.

2.4.6 Confusion Matrix

Confusion Matrix is a performance measurement for machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values.



The confusion matrix offers a comprehensive view of the model's performance, showing the number of true positives (correctly predicted delays), false negatives (delays that were not predicted), true negatives (correctly predicted on-time flights), and false positives (on-time flights predicted as delayed). For airlines, this detailed breakdown is invaluable. It helps identify specific areas where the model excels or needs improvement. For instance, the confusion matrix might reveal that the model performs well under certain weather conditions but not others, or during specific times of the day. Such insights are crucial for targeted improvements and strategic decision-making in operations.

2.5 Ethical Considerations and Limitations

2.5.1 Ethical Aspects:

Ensured that the data used was anonymized and did not contain sensitive personal information about passengers or airline staff, adhering to privacy and ethical standards.

2.5.2 Methodological Limitations:

Acknowledged the limitations of using historical data, which might not fully capture current trends or future patterns.
Discussed the potential biases in the data and the limitations inherent in the chosen algorithms, such as overfitting in tree-based models.

3. Analysis

3.1 Data Cleaning

3.1.1 Remove duplicates:

We start by identifying and removing duplicate records, which can skew results and lead to inaccurate models.

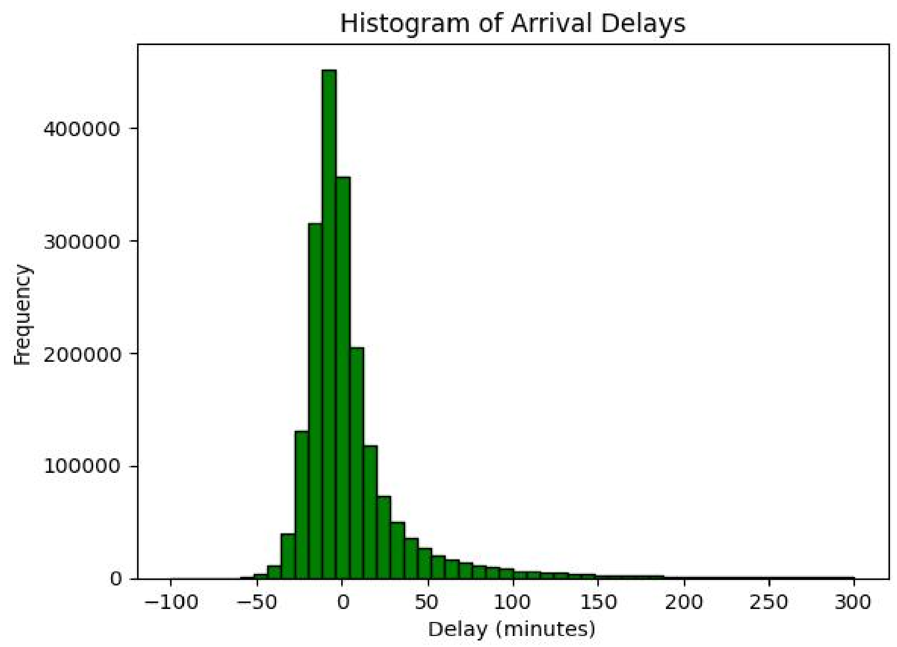
3.1.2 Handling missing values:

Drop columns with more than 1 million (>50%) missing values, these columns are much less meaningful to predict the target.
Remove rows that have missing values in the core column: ARRIVAL_DELAY, there are 53364 out of 2000000 rows without ARRIVAL_DELAY value.

3.1.3 Check data distribution:

Check data distribution using pd.describe(), it can check mean, standard deviation, min and max values, quantile numbers for each column. We drop columns with very little or no variation because they contain little information value and can unnecessarily increase the complexity of the model without adding to its performance.

We also plot histograms using matplotlib, determine how to split delayed and not delayed groups and find most relevant variables.

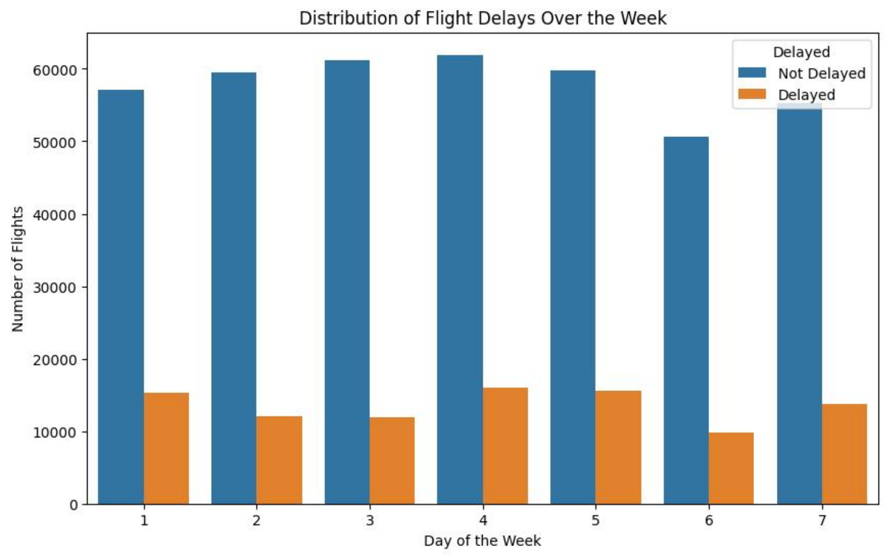


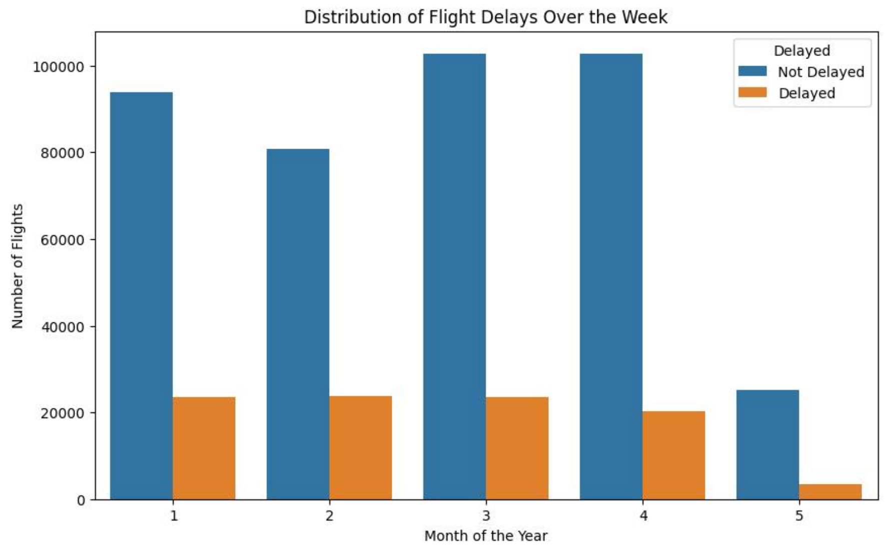
Plot the histogram of ARRIVAL_DELAY, the delay has a long tail on the right, which means most of the flights are on time, but delayed group also takes a quite large proportion.
We consider that a flight is delayed when it arrives 15 minutes later than the scheduled time, using the standard from FAA (Federal Arbitration Act). There are 18.89% records identified as delayed.

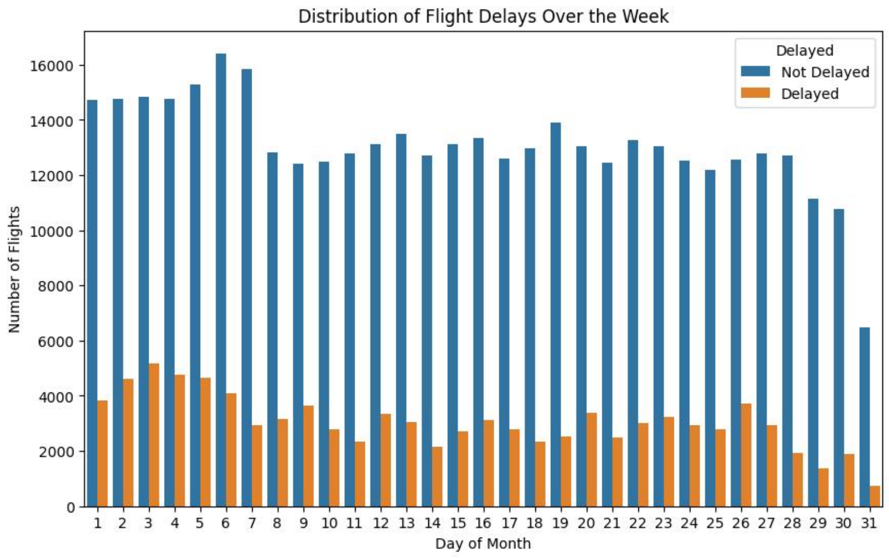
3.2 Feature Engineering

This step involves creating new features or modifying existing ones to improve model performance.

3.2.1 Create bar charts to visually analyze the frequency of delays across different time scales:







We could see there is some data variation based on time scales. On certain days, the number of flights and their associated delays are significantly lower compared to other days, like the last 3 days of the month and Saturday. But there is not a very clear trend or cyclical pattern, we could still consider the day of week, month, and day of the month as our features, since there possibly exist hidden patterns and interaction effects.

3.2.2 Statistical tests to check significance of categorical variables:

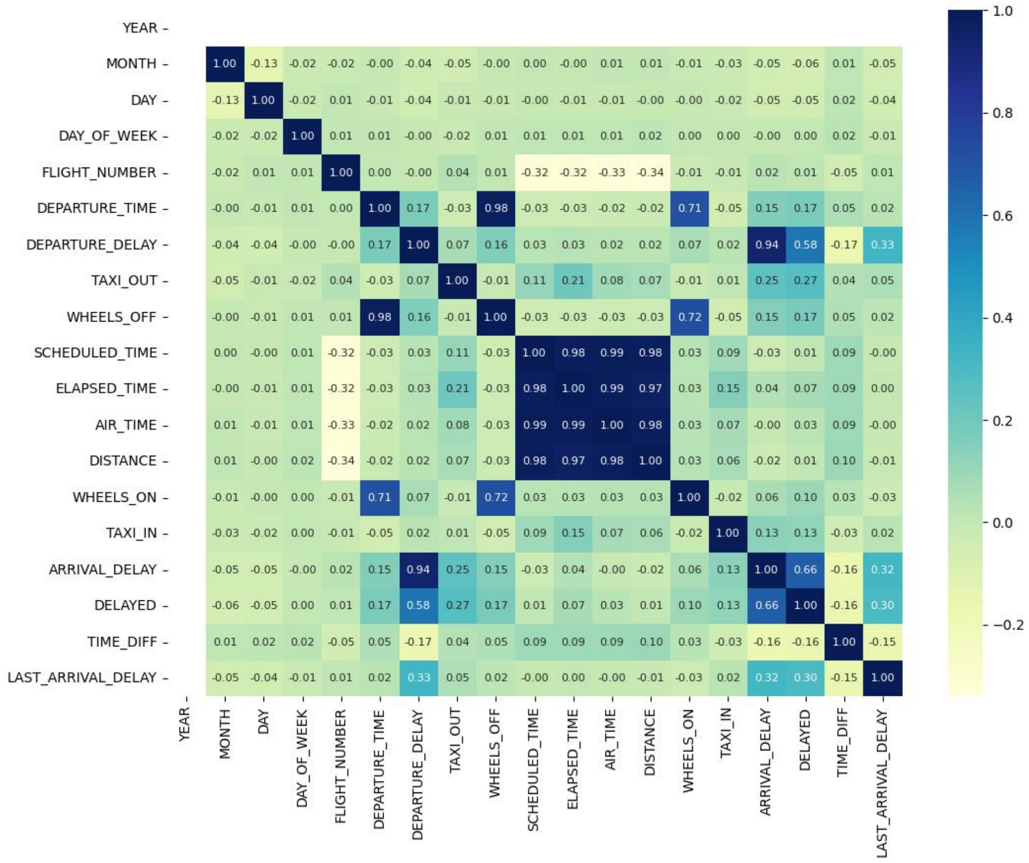
The Chi-squared (χ^2) test, especially in the context of feature selection for machine learning models, is used to determine whether there is a significant association between two categorical variables. It's a useful tool for deciding which variables might be relevant for inclusion in the model.

The p-value in Chi test tells us whether there is a significant association between two variables, if the p-value is less than 0.05, we can conclude that two variables are relevant to each other.

We test the relationship between delayed and day of the week, month, day, airline, in airport, destination airport, and tail number, which all have a p-value equals to 0.0, showing they have a significant association with delayed.

3.2.3 Statistical correlation exploration:

Checking Pearson correlation coefficients for all variables in the dataset is a common practice in data preprocessing. The correlation table could help us measure the linear relationship between two variables. In machine learning, especially in linear regression models, highly correlated independent variables can distort the result, which is called multicollinearity. Identifying these variables using a correlation table is necessary. Understanding the correlation could also extract insights for feature selection, selecting variables which are highly correlated to delayed can simplify the model and make it easier to interpret.



In our data, there are some variables such as elapsed time, airtime, scheduled time that are highly correlated with each other, and they are not useful in our prediction method since we want to predict whether the flight delayed before the flight took off, therefore we can drop these variables. Lastly, we drop taxi out, wheels off, elapsed time, scheduled time, airtime, wheels on, taxi in, scheduled arrival, arrival time, arrival delay.

3.2.4 Create new feature:

Initially, we trained the model using only the original features and observed that without including the 'departure delay' variable, the models' ROC scores were relatively low, barely surpassing 0.6, which is only slightly better than random guessing. When we added the variable

departure delay into the model, the score jumped up to 0.94, but this variable became the most important variable and other variables became meaningless. Using this variable also makes no sense in our prediction scenario, we want to predict the flight arrival delay before the flight took off so that departure delay does not satisfy our needs.
After looking through domain knowledge and other data competition cases, we created two new features "last arrival delay" and "time diff". Last arrival delay means the delay time for the first leg of a connecting flight, time diff is the time difference (minutes) between the first and second leg of a connecting flight. The definition of connecting flight is the same flight (same tail number) that reaches the destination through two or more flights, the first journey's destination airport is the second journey's origin airport, and all journeys happen in the same day. These two variables can be calculated based on original variables. We believe that these two variables directly influence the likelihood of delays in connecting flights.

3.3 Feature Standardization

Finally, our features are:

Name	Type	Definition
MONTH	int	month
DAY	int	Day of the month
DAY OF WEEK	int	Day of the week
AIRLINE	object	Airline identifier
FLIGHT_NUMBER	int	Flight identifier
ORIGIN_AIRPORT	object	Starting airport
DESTINATION_AIRPORT	object	Destination airport
SCHEDULED_DEPARTURE	int	Planned departure time (minutes)
LAST_ARRIVAL_DELAY	int	Delay time for the first leg of a connecting flight
TIME_DIFF	int	Time difference (minutes) between the first and second leg of a connecting flight

Independent variable: Whether the flight was delayed when arriving. 1 is delayed, 0 is not delayed.

3.3.1 Handle missing values

Fill all missing values with the mean value of that column since some machine learning models require complete datasets.

3.3.2 Numeric features normalization

Use StandardScaler() function to normalize features by removing the mean and scaling to unit variance. The process can be mathematically represented as $z = \frac{(x-\mu)}{\alpha}$. where x is the original feature value, μ is the mean of the feature, α is the standard deviation of the feature. It can help algorithms perform better and converge faster when features are on a relatively similar scale and are normally distributed. Moreover, logistic regression requires standardized data for effective training.

3.3.3 Encoding categorical variables

Use OneHotEncoder() function to create a binary column for each category of a categorical variable since many machine learning algorithms require input to be numerical since they perform operations on numbers, and it is particularly useful as it does not impose any ordinality of non-ordinal features.

3.4 Model Training

3.4.1 Data splitting

We split the dataset into train and test data, 75% is train data and 25% is test data. The primary purpose of splitting data is to have a reliable way to assess the performance of a model. The model is trained on the training set and then tested on the unseen data of the test set. This helps in evaluating how well the model generalizes to new, unseen data. Secondly it can prevent overfitting, overfitting occurs when a model learns not only the underlying patterns in the training data but also its noise. Such a model performs well on the training data but poorly on new, unseen data. By having a separate test set, you can check for overfitting and ensure that the model is robust and generalizes well.

3.4.2 Parameters optimization

We use grid search to explore a specified range of values for each hyperparameter. It's a brute-force approach that creates and evaluates models for every possible combination of hyperparameters in the specified grid. The main goal of grid search is to identify the combination of parameters that yields the best model performance

3.4.3 Models performance

We use ROC score as our core evaluation index.
Below is the result table:

Model	Train Data Performance	Test Data Performance
Logistic Regression	0.751	0.754
Decision Tree	0.792	0.796
Random Forest	0.782	0.783
XGBoost	0.845	0.839
LightGBM	0.841	0.843

It is quite clear that ensemble learning models show significantly higher performance, XGBoost performing slightly better on the training data (0.845 vs. 0.841) and LightGBM performing slightly better on the test data (0.843 vs. 0.839). The performance of the models on the test data is close to their performance on the training data, indicating good generalization. LightGBM has almost identical performance on training and test data (0.841 and 0.843, respectively), which suggests it generalizes well in this scenario. There is no significant sign of overfitting or underfitting for any of the models, as the performance on the training and test datasets is quite close.

4. Conclusion & Future Work

4.1 Weaknesses and Limitations

4.1.1 Data Limitations:

- Historical Data Bias: Our project heavily relied on historical flight data from 2015, which may not accurately represent current or future trends due to changes in airline operations, traffic patterns, or technological advancements.
- Incomplete Variables: Certain influential factors like real-time weather updates, unexpected technical issues, or air traffic control directives were not included in our dataset, potentially limiting the comprehensiveness of our analysis.
- Generalization Challenges: The models might not generalize well to all types of flights or airports, since the data only contains information for domestic U.S. flights information.

4.1.2 Methodological Limitations:

- Hyperparameter Tuning and Model Selection
  The process of hyperparameter tuning (grid search) and model selection can lead to models that are overly optimized on the specific dataset used. This might not translate well to other datasets or real-world applications.
- Model Complexity
  More complex models like XGBoost and LightGBM have higher scores, but this complexity can come with a cost. Complex models require more computational resources and are more difficult to interpret. There's also the risk that they might be capturing noise in the data rather than underlying patterns.
- Computer Memory Constraint
  Due to the availability of system memory, we gave up a half proportion of the original data samples and did not include more features such as the number of delayed flights on the same day before the flight took off, which requires a huge number of rows joining operation. These limitations affect the overall performance of our models.

4.1.3 Scope of Study:

- Focus on Predictive Accuracy: The project primarily focused on predictive accuracy, which might have overshadowed other important aspects like interpretability and the practical applicability of the predictions.
- Lack of Real-World Testing: The models were not tested in a real-world operational setting, which could reveal additional challenges and practical considerations not accounted for in a controlled analysis environment.

4.2 Avenues for Future Research

4.2.1 Expanding Data Sources:

- Real-Time Data Integration: Future research could incorporate real-time data, such as weather changes, live air traffic updates, and on-the-day operational changes, to enhance the predictive model's responsiveness and accuracy.

- Broader Time Frame: Analyzing data over a more extended period or including more recent data could provide insights into evolving trends and improve the models' adaptability to current operational realities.

4.2.2 Methodological Improvements:

- Exploring New Algorithms: Investigating emerging machine learning algorithms or developing custom models tailored to the specific nuances of flight operations could yield better predictive performance.
- Interdisciplinary Approaches: Combining machine learning with other disciplines like operations research or human factors could offer a more holistic approach to predicting and managing flight delays.

4.2.3 Operational and Practical Applications:

- Real-World Implementation and Testing: Piloting the models in real operational settings would provide valuable feedback on their practical effectiveness and areas for improvement.
- User-Centered Design: Future studies could focus on how flight delay predictions are communicated and used by airlines, airport staff, and passengers, ensuring that the information is actionable and user-friendly.

4.2.4 Broader Scope of Study:

- Economic and Environmental Impact: Investigating the economic implications of flight delays and the potential environmental impact of optimized flight schedules could add valuable dimensions to the research.
- Global and Regional Analysis: Expanding the study to include global or region-specific analysis could uncover unique challenges and solutions applicable in different geographical contexts.

4.3 Personal Learning

- **Yedian Cheng:** In this project, I gained substantial knowledge about machine learning and statistics, particularly in data cleaning and analysis, understanding correlations in data to select appropriate labels, and learning to build and tune models. This experience served as an entry point into the realm of machine learning and sparked a deeper interest in the field. The practical skills I developed in this project have immediate applications in my academic and professional future. I can now approach more advanced courses in data science and machine learning with confidence, leveraging the hands-on experience in model building and data analysis I've acquired. Post-graduation, these skills are directly transferable to the workplace, especially in industries that rely heavily on data-driven decision making.
- **Tianzi Qin:** During the project, I found that data quality is very important, we spent a lot of time cleaning data and ensuring data integrity before building reliable models. Secondly, feature engineering also plays a crucial role, it requires domain knowledge for this specific industry, and one good feature could improve the model performance dramatically. Lastly, working with different models taught me about the strengths and limitations of each. I learned to appreciate that no single model is universally best, and

that model selection should be based on the specific nature of the data and the problem at hand.

- **Baozhang Min:** While data analysis felt routine in my past learning and work, exploring machine learning for our project turned out to be more enlightening than I expected. It introduced diverse approaches to tackle multifaceted problems, such as using the Pearson Correlation Coefficient. This powerful yet straightforward tool enhances accuracy in significantly less time by identifying key variables efficiently. As I immersed myself in algorithms during my computer science studies, the experience demystified math, revealing its profound link to real-world problem-solving. Despite my initial worries about complicated notations, this project expanded my perspective on problem-solving. Now, I can totally envision the crucial role that algorithms and math play in constructing effective product development.

5. References

https://www.kaggle.com/datasets/usdot/flight-delays/data
https://en.wikipedia.org/wiki/Logistic_regression
https://scikit-learn.org/stable/modules/tree.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
https://en.wikipedia.org/wiki/Random_forest
https://www.geeksforgeeks.org/xgboost/
https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/
https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

6. Presentation Video Link:

https://northeastern-my.sharepoint.com/:v:/g/personal/qin_tianz_northeastern_edu/EfXxMQ0ryBNKr0ETLRxWrQoBVrwbmRHq9uqXzvfbWG4Zzw