

Name: Umanga Niroula

ID: 14825284

Submitted To: Siddharth Neupane

**Predicting Flight Delays using PySpark and Machine Leaning**

STW7082CEM Big Data Management and Data

Visualisation

Table of Contents

[1. Introduction 4](#_Toc170330920)

[2. Dataset and Data Analysis 5](#_Toc170330921)

[2.1. Overview of Dataset 5](#_Toc170330922)

[2.1.1. Sample Data 5](#_Toc170330923)

[2.2. Setting Up Spark Environment 6](#_Toc170330924)

[2.3. Basic Data Exploration 7](#_Toc170330925)

[2.4. Data Pre-processing 8](#_Toc170330926)

[2.4.1. Data Cleaning 8](#_Toc170330927)

[2.4.2. Column and Data manipulation 9](#_Toc170330928)

[2.4.3. Assembling columns 11](#_Toc170330929)

[2.5. Machine Learning Models 12](#_Toc170330930)

[2.5.1. Decision Tree Model 13](#_Toc170330931)

[2.5.2. Logistic Regression 18](#_Toc170330932)

[2.6. Explorative Analysis and Visualizations 21](#_Toc170330933)

[3. Discussion 27](#_Toc170330934)

[3.1. Decision Tree Model 27](#_Toc170330935)

[3.2. Logistic Regression Model 27](#_Toc170330936)

[3.3. Model Comparison and Future Work 28](#_Toc170330937)

[4. Conclusion 29](#_Toc170330938)

[5. Reference 30](#_Toc170330939)

[6. Appendix 31](#_Toc170330940)

**List of Figures**

[Figure 1: Setting up PySpark Enviroment. 6](#_Toc170330868)

[Figure 2: Loading Dataset to Data Frame. 7](#_Toc170330869)

[Figure 3: Displaying the Total Number of Rows in the Dataset. 7](#_Toc170330870)

[Figure 4: Top 5 Rows of the Dataset. 8](#_Toc170330871)

[Figure 5: Dropping Flight Column. 9](#_Toc170330872)

[Figure 6: Removing Missing Values from the Record. 9](#_Toc170330873)

[Figure 7: Converting Miles to Km. 10](#_Toc170330874)

[Figure 8: Creating a Label Column to Classify weather a Flight is Delayed or Not. 10](#_Toc170330875)

[Figure 9: Converting Categorical to Numerical Value. 10](#_Toc170330876)

[Figure 10: Dataset after Data Pre-processing. 11](#_Toc170330877)

[Figure 11: Combining Predictor Columns. 11](#_Toc170330878)

[Figure 12: Output of Combining Predictor Columns. 12](#_Toc170330879)

[Figure 13: Splitting Data into Train and Test Data. 13](#_Toc170330880)

[Figure 14: Training the Model using Decision Tree Model. 13](#_Toc170330881)

[Figure 15: Prediction using Decision Tree Model on Test Data. 14](#_Toc170330882)

[Figure 16: Prediction vs. Probability for Decision Tree Model. 14](#_Toc170330883)

[Figure 17: Calculating the Accuracy of Decision Tree Model. 16](#_Toc170330884)

[Figure 18: Confusion Matrix. 17](#_Toc170330885)

[Figure 19: Accuracy of Decision Tree Model. 18](#_Toc170330886)

[Figure 20: Training the Logistic Regression Model. 18](#_Toc170330887)

[Figure 21: Predicting the Test Data using Logistic Regression Model. 18](#_Toc170330888)

[Figure 22: Calculating Precision and Recall of Logistic Regression Model. 19](#_Toc170330889)

[Figure 23: Confusion Matrix for Logistic Regression Model. 19](#_Toc170330890)

[Figure 24: Precision and Recall of Logistic Regression Model. 20](#_Toc170330891)

[Figure 25: Calculating AUC. 20](#_Toc170330892)

[Figure 26: Distribution of Flight Delays Visualization. 21](#_Toc170330893)

[Figure 27: Delay of Flight by Airport Visualization. 22](#_Toc170330894)

[Figure 28: Average Delay of Flight by Carrier Visualization. 23](#_Toc170330895)

[Figure 29: Total Delay of Flight per Month. 24](#_Toc170330896)

[Figure 30: Average Delay of Flight by Carrier per Month. 25](#_Toc170330897)

[Figure 31: Flight Duration vs. Average Delay Visualization. 26](#_Toc170330898)

[Figure 32: Installing PySpark. 31](#_Toc170330899)

[Figure 33: Successful Installation of PySpark. 31](#_Toc170330900)

# Introduction

In this report, we explore the application of machine learning techniques to predict flight delays using PySpark, a powerful framework for distributed data processing. Flight delays pose significant challenges for airlines, airports, and passengers, making predictive models valuable tools for improving operational efficiency and passenger experience. By leveraging historical flight data, we aim to build and evaluate a classification model to determine whether a flight will arrive late (Ball et al., 2024).

The dataset used in this analysis, sourced from Kaggle, includes various features such as the month, day of the month, day of the week, carrier information, flight number, origin airport, flight distance (in miles), scheduled departure time, flight duration, and the actual delay time. The primary objective is to predict the delay status of flights, classifying them as either delayed or on-time (Kaggle, n.d.). Furthermore, we employ a Logistic Regression model within a pipeline that includes several stages for data preprocessing and feature engineering. The pipeline includes transforming categorical features, assembling feature vectors, scaling numeric features, and finally, training the logistic regression model. We also utilize cross-validation to tune hyperparameters and optimize the model.

Likewise, the performance of the model is evaluated using key metrics such as precision, recall, F1 score, and the Area Under the ROC Curve (AUC). These metrics help us understand the model's ability to accurately predict flight delays and its overall robustness (Sokolova and Lapalme, 2009). Hence, through this analysis, we aim to demonstrate the effectiveness of using PySpark for large-scale machine learning tasks and provide insights into improving flight delay predictions using advanced data processing and modeling techniques.

# Dataset and Data Analysis

## Overview of Dataset

The dataset used in this analysis, sourced from Kaggle, provides comprehensive details about various flights, including the factors that could potentially influence delays (Kaggle, n.d.). There are all together 10 columns and 275001 rows which qualifies this data for big data analysis. Below is a summary of the dataset's key features:

* **mon**: Month of the flight (e.g., 1 for January, 2 for February)
* **dom**: Day of the month when the flight occurred
* **dow**: Day of the week (e.g., 1 for Monday, 2 for Tuesday)
* **carrier**: The airline carrier code (e.g., OO for SkyWest Airlines, B6 for JetBlue Airways)
* **flight**: Flight number assigned by the airline
* **org**: Origin airport code (e.g., ORD for Chicago O'Hare, JFK for John F. Kennedy)
* **mile**: Distance of the flight in miles
* **depart**: Scheduled departure time (in 24-hour format)
* **duration**: Scheduled flight duration in minutes
* **delay**: Actual delay in minutes (negative values indicate early arrivals, positive values indicate delays)

### Sample Data

Following table contains the top 10 rows of the dataset.

Table 1: Dataset Sample.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **mon** | **dom** | **dow** | **carrier** | **flight** | **org** | **mile** | **depart** | **duration** | **delay** |
| 10 | 10 | 1 | OO | 5836 | ORD | 157 | 8.18 | 51 | 27 |
| 1 | 4 | 1 | OO | 5866 | ORD | 466 | 15.50 | 102 | NA |
| 11 | 22 | 1 | OO | 6016 | ORD | 738 | 7.17 | 127 | -19 |
| 2 | 14 | 5 | B6 | 199 | JFK | 2248 | 21.17 | 365 | 60 |
| 5 | 25 | 3 | WN | 1675 | SJC | 386 | 12.92 | 85 | 22 |
| 3 | 28 | 1 | B6 | 377 | LGA | 1076 | 13.33 | 182 | 70 |
| 5 | 28 | 6 | B6 | 904 | ORD | 740 | 9.58 | 130 | 47 |
| 1 | 19 | 2 | UA | 820 | SFO | 679 | 12.75 | 123 | 135 |
| 8 | 5 | 5 | US | 2175 | LGA | 214 | 13.00 | 71 | -10 |

## Setting Up Spark Environment

Firstly, we set up Spark using PySpark libraries to handle big data processing and analysis. We configure a local Spark cluster to make use of all CPU cores, making computation more efficient. We create a SparkSession object to work with Spark features, allowing us to manipulate data and perform machine learning tasks (Apache Spark, n.d.). We also double-check the installed Spark version to ensure it works well with our code, which is done in the following figure 1.

****

Figure 1: Setting up PySpark Enviroment.

## Basic Data Exploration

Now, we load the dataset “flights-larger.csv” into a Spark Data Frame “flights\_df” to explore and analyze it, as shown in the following figure 2.

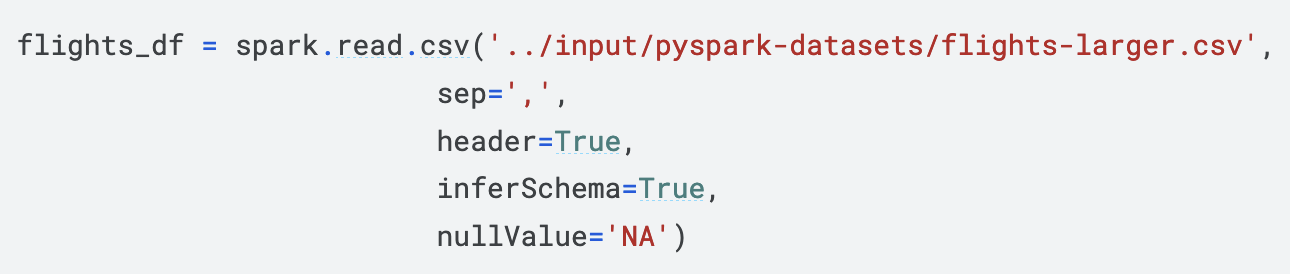


Figure 2: Loading Dataset to Data Frame.

Initial exploration tasks help us understand the dataset's structure and characteristics, which is why we examine the number of records and the data types of columns to gain insights into the dataset's composition as shown in the following figure 3.

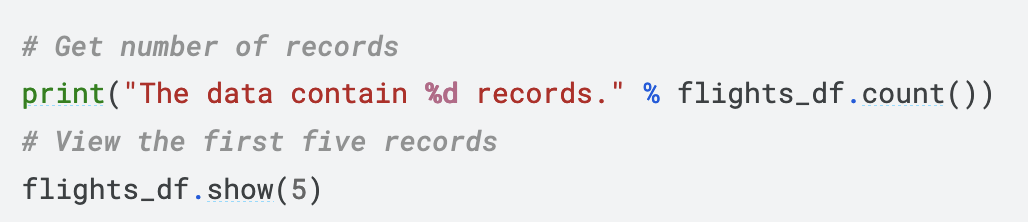


Figure 3: Displaying the Total Number of Rows in the Dataset.

Additionally, we display the first five records of the dataset to get a sense of its format and content as shown in the following figure 4.

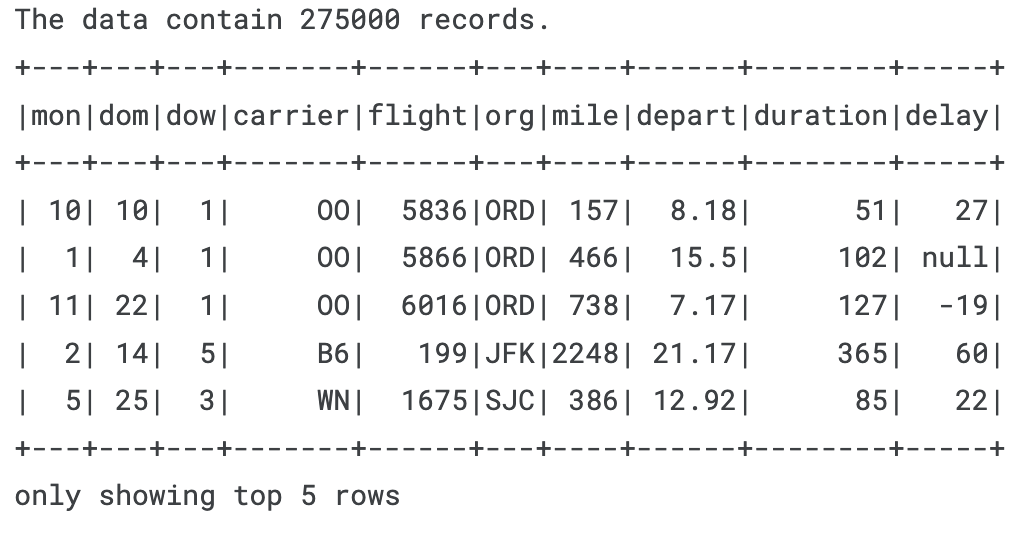


Figure 4: Top 5 Rows of the Dataset.

## Data Pre-processing

In this section, we outline the steps for preparing the data to train our machine learning model. The data pre-processing includes cleaning the data, manipulating columns and data, and assembling the columns into a feature vector for the model.

### ****Data Cleaning****

The initial steps in data cleaning involve removing unnecessary columns and handling missing values. Following are the techniques which is applied for data cleaning:

* **Removing a flight column:**

The flight column, which is not useful for our analysis, is dropped from the dataset. The flight column usually contains a unique identifier for each flight, like a flight number. While this is useful for tracking flights, it doesn't provide meaningful information about the factors affecting delays. Including this column in our model wouldn't help predict flight delays and could add unnecessary noise to the dataset.



Figure 5: Dropping Flight Column.

If we include a unique identifier like a flight number in the training data, the model might memorize specific flights instead of learning general patterns. This can cause overfitting, meaning the model will do well on the training data but poorly on new, unseen data.

* **Removing rows with missing values:**

All rows containing any missing values are removed to ensure the dataset is complete

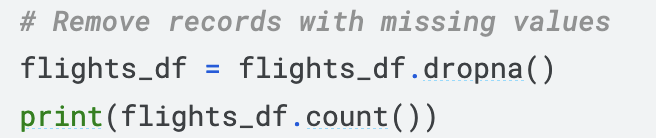


Figure 6: Removing Missing Values from the Record.

### Column and Data manipulation

Next, we manipulate the columns to create a new label column and convert categorical data into numerical values. Following are the details about column and data manipulation:

* **Converting distances from miles to kilometers**

The mile column is converted to km and then the original mile column is dropped. Kilometers are a standard unit of measurement in most parts of the world, as a result having a uniform unit of measurement for distance can simplify the analysis and make the dataset more homogeneous. Further, the original mile column is dropped as it becomes redundant.

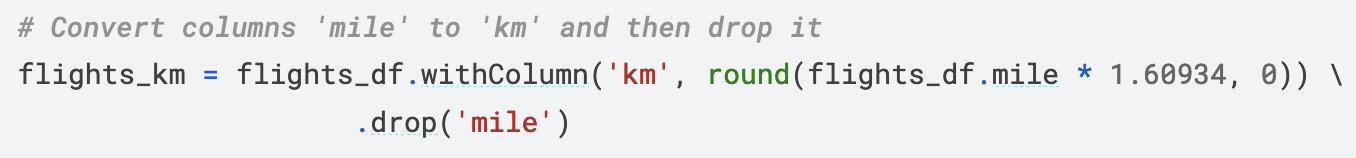


Figure 7: Converting Miles to Km.

* **Creating a new label column**

A new boolean column label is created to indicate whether a flight is delayed. A flight is considered delayed if it arrives 15 minutes or more after its scheduled time (Bureau of Transportation Statistics, n.d.). By creating a label column, we establish a binary classification problem where the goal is to predict whether a flight is delayed (1) or not (0). According to the FAA's definition, a flight is considered delayed if it arrives 15 minutes or more after its scheduled time. Using this threshold, we create a clear and standardized criterion for what constitutes a delay.

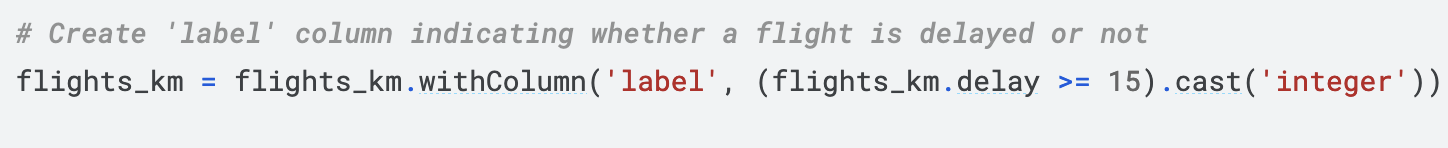


Figure 8: Creating a Label Column to Classify weather a Flight is Delayed or Not.

* **Converting categorical columns to numerical values**

The carrier and org columns, which hold categorical data, are converted to indexed numerical values using the StringIndexer. Most machine learning algorithms cannot directly handle categorical data. They require input data to be in numerical format.Following figure 9 shows how the conversation is done through python.

****

Figure 9: Converting Categorical to Numerical Value.

After the completion of column and data manipulation following figure 10 shows how the dataset looks like.

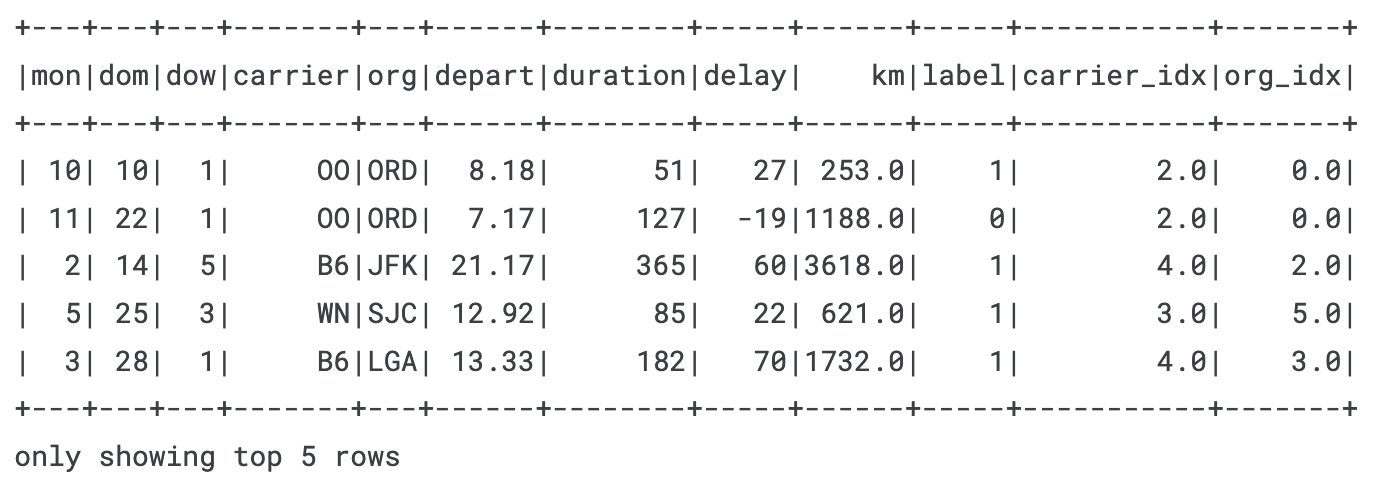
****

Figure 10: Dataset after Data Pre-processing.

### Assembling columns

* **Creating a feature vector**

A VectorAssembler is used to combine the predictor columns into a single features column, which will be used for model training. In Spark's MLlib and other frameworks, several machine learning algorithms require the input data to be in a specific format. All predictor variables, also known as features, should be merged into a single vector column. This format is crucial for algorithms like logistic regression, decision trees, and support vector machines. Following figure 11 shows how the combination is done.

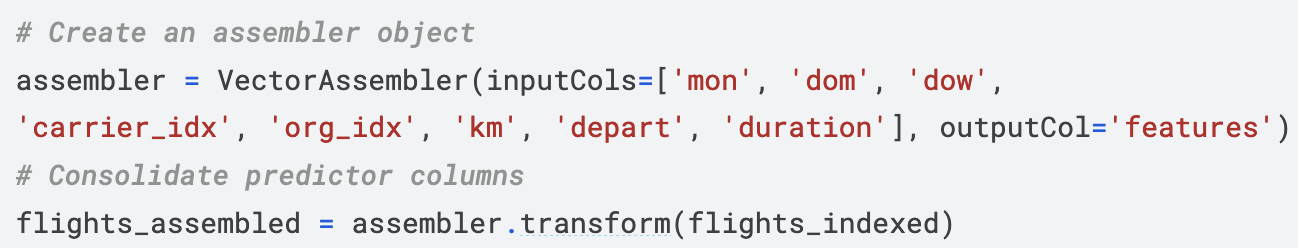
****

Figure 11: Combining Predictor Columns.

* **Checking the resulting column**

The new features column is verified to ensure it has been correctly assembled as shown in the following figure 12.

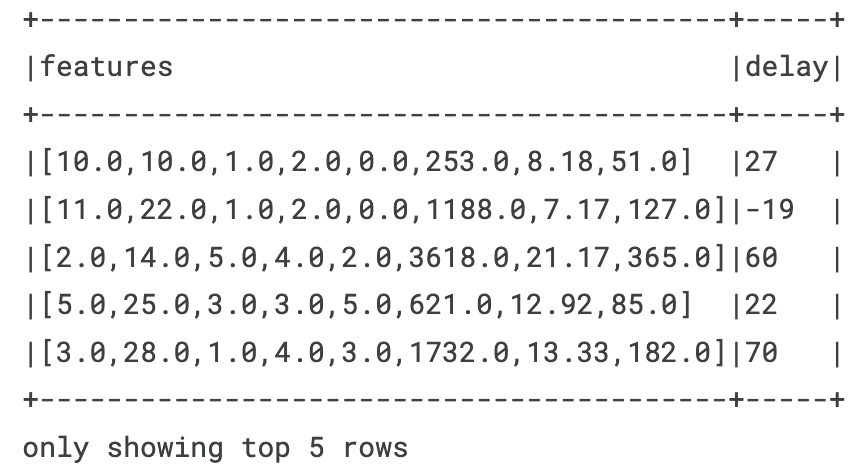
****

Figure 12: Output of Combining Predictor Columns.

## Machine Learning Models

Before training a machine learning model, it's essential to split the dataset into two parts: a training set and a testing set. This is crucial for evaluating the model's performance on unseen data, ensuring it generalizes well. If we trained and tested the model on the same data, the model might simply memorize the training data, performing well on it but poorly on new data. This is known as overfitting. Splitting the data allows us to validate the model's performance on data it hasn't seen before, providing a more accurate assessment of its predictive power. Following figure 13 shows how the dataset is split into train and test data.

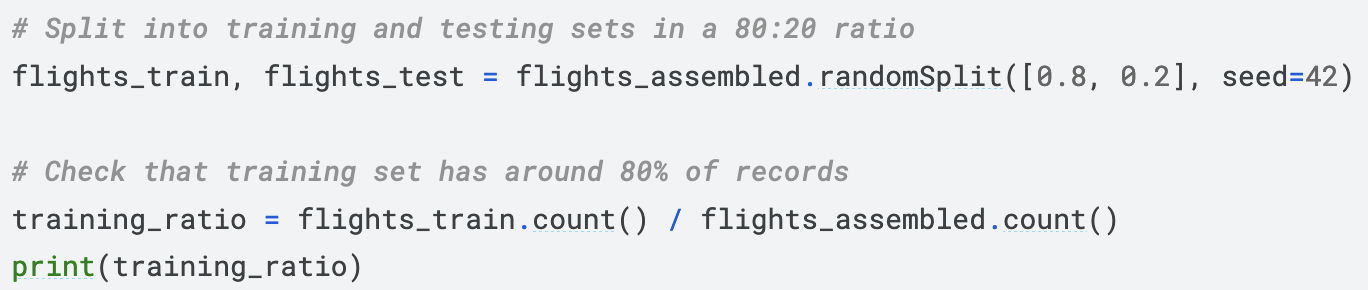


Figure 13: Splitting Data into Train and Test Data.

The above figure 13 shows the dataset flights\_assembled is split into training (80%) and testing (20%) sets using the randomSplit method. The seed ensures reproducibility. The training set ratio is then calculated and printed to verify the split.

### Decision Tree Model

We consider using a decision tree because it is straightforward to understand and interpret. Decision trees visualize the decision-making process, making it easy to see how the model arrives at its predictions (Breiman, Friedman, Olshen, & Stone, 1984). For example, it's clear which features are important and how the model splits the data based on those features. Additionally, decision trees can capture non-linear relationships between features and the target variable, which is useful when the relationship between flight delays and predictors (such as departure time and day of the week) is complex. Below are the steps illustrating how the decision tree model has been implemented:

* **Model Training**

In this section, we create and train a Decision Tree classifier using the training dataset ash shown by the following figure 14.

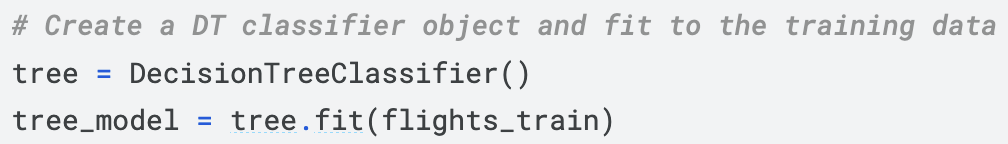


Figure 14: Training the Model using Decision Tree Model.

* **Model Prediction**

After training the Decision Tree model, the next step is to use it to make predictions on the test dataset as shown in the following figure 15.

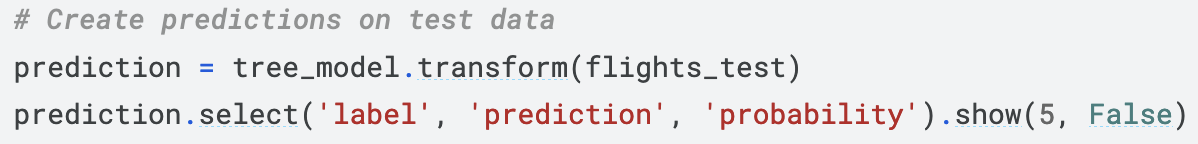


Figure 15: Prediction using Decision Tree Model on Test Data.

The output illustrated in the following figure 16, shows where the model's predictions differ from the actual outcomes, highlighting areas for improvement. For example, the first row shows a misclassification where the actual status was on-time (label = 0) but the model predicted a delay (prediction = 1.0).

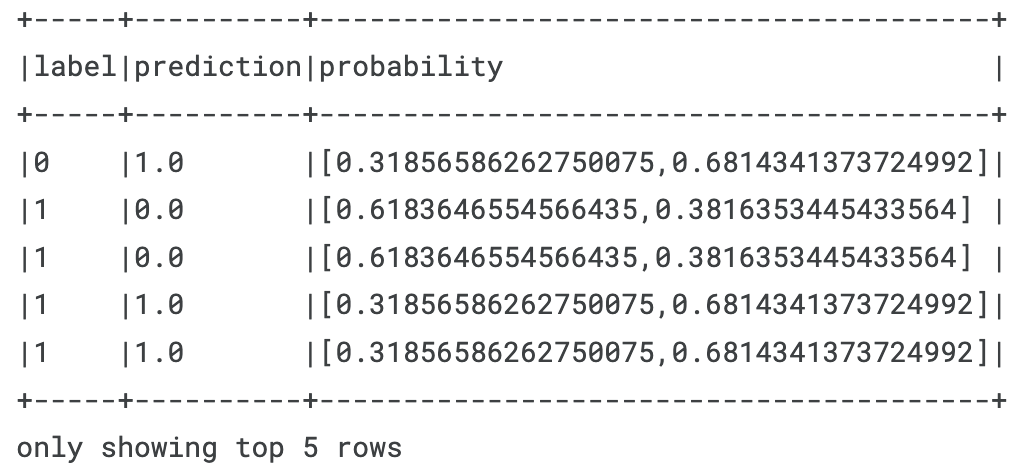


Figure 16: Prediction vs. Probability for Decision Tree Model.

Similarly, the probability scores give extra context, showing how confident the model is in its predictions. This helps in understanding and interpreting the model's performance. In the first row, the model was 68.14% confident in predicting a delay, even though the actual status was on-time.

* **Model Evaluation**

To evaluate the model, we generate a confusion matrix that provides a detailed breakdown of prediction outcomes. A confusion matrix gives a useful breakdown of predictions versus known values. It has four cells which represent the counts of: True Negatives (TN) — prediction is negative & label is negative that are explained briefly below:

* False Positive (FP): Prediction is positive & label is negative.
* True Positives (TP): Prediction is positive & label is positive.
* False Negatives (FN): Prediction is negative & label is positive.
* False Positives (FP): Prediction is positive & label is negative.

Using these four measure, we can then calculate the accuravy of the model as follows:

**Accuracy=(TN+TP)/(TN+TP+FN+FP)**

**Now, to calculate the accuracy, code shown in the following figure 17 has been used.**

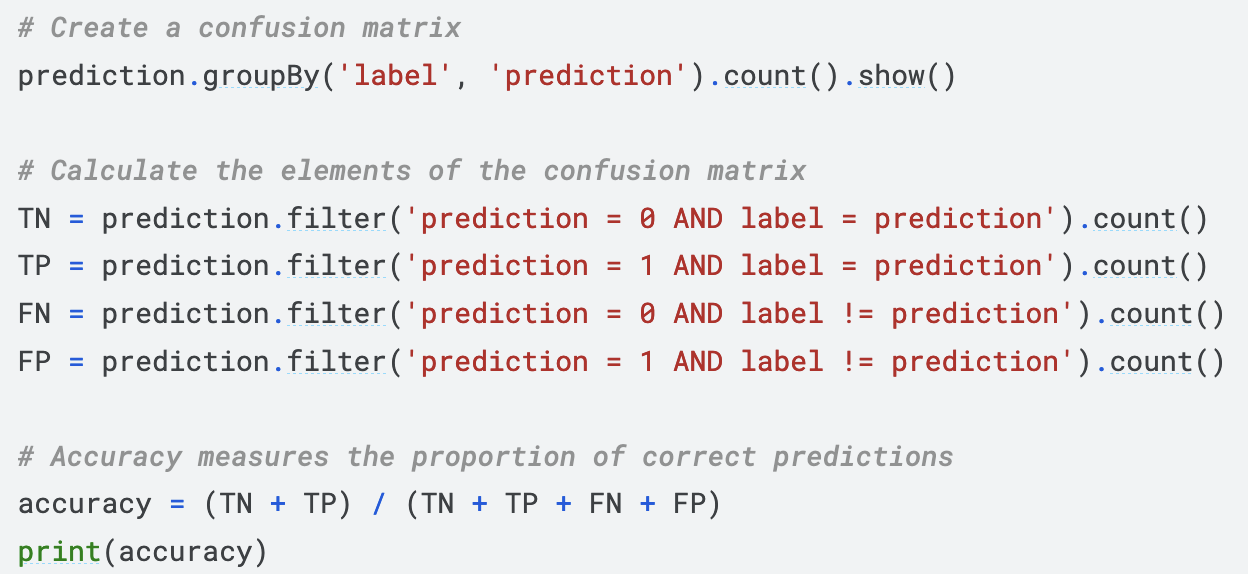


Figure 17: Calculating the Accuracy of Decision Tree Model.

* **Calculating Confusion Matrix Elements**

Next, we calculate the individual elements of the confusion matrix: True Negatives (TN), True Positives (TP), False Negatives (FN), and False Positives (FP). Following is the explanation of all the elements of confusion matrix:

* + **True Negatives (TN)**: Cases where the model correctly predicted no delay.
  + **True Positives (TP)**: Cases where the model correctly predicted a delay.
  + **False Negatives (FN)**: Cases where the model predicted no delay but the flight was delayed.
  + **False Positives (FP)**: Cases where the model predicted a delay but the flight was on-time.

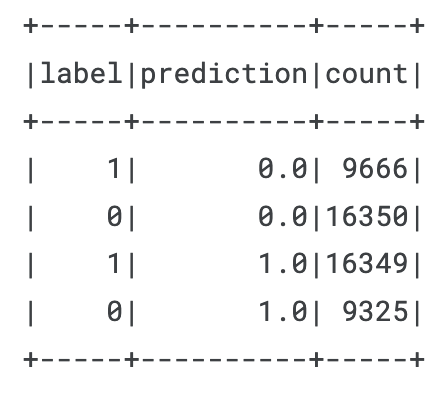


Figure 18: Confusion Matrix.

To evaluate the model, we generate a confusion matrix that provides a detailed breakdown of prediction outcomes. The confusion matrix includes the following elements:

* + **False Negatives (FN):** 9,666 instances where the model predicted no delay (0), but the actual label was a delay.
  + **True Negatives (TN):** 16,350 instances where the model correctly predicted no delay (0) and the actual label was also no delay.
  + **True Positives (TP):** 16,349 instances where the model correctly predicted a delay (1) and the actual label was also a delay.
  + **False Positives (FP):** 9,325 instances where the model predicted a delay (1), but the actual label was no delay.
* **Calculating Model Accuracy**

Now we calculate the accuracy, the model achieved an accuracy of approximately 63.26%. This metric indicates that the model correctly predicted the delay status of flights about 63.26% of the time. Despite the reasonable accuracy, the confusion matrix reveals a significant number of false predictions (both false negatives and false positives), indicating areas where the model may need improvement.



Figure 19: Accuracy of Decision Tree Model.

### Logistic Regression

After evaluating the Decision Tree model, we explore Logistic Regression as an alternative classification algorithm. Logistic Regression is a popular and straightforward method for binary classification problems. It models the probability that a given input point belongs to a certain class by using a logistic function (Hosmer Jr, Sturdivant, & Lemeshow, 2013). This makes it well-suited for our task of predicting flight delays.

* **Model Training**

We train the Logistic Regression model on the flights\_train dataset. The training code is shown in the following figure 20.



Figure 20: Training the Logistic Regression Model.

* **Model Prediction**

Once the model is trained, we use it to make predictions on the test dataset (flights\_test). The predictions and their evaluation are shown by the following figure 21.

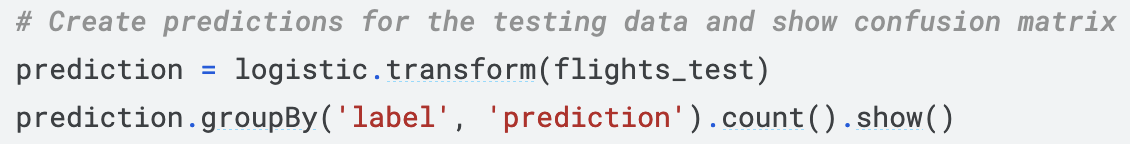


Figure 21: Predicting the Test Data using Logistic Regression Model.

* **Model Evaluation**

The evaluation metrics for the Logistic Regression model are calculated using the precision and recall as shown in the following figure 22.

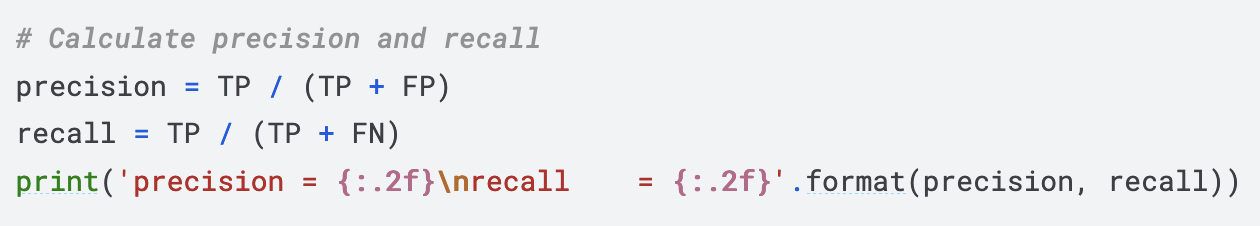


Figure 22: Calculating Precision and Recall of Logistic Regression Model.

* **Calculating Confusion Matrix Elements**

Following figure 23 shows the calculation of confusion matrix.

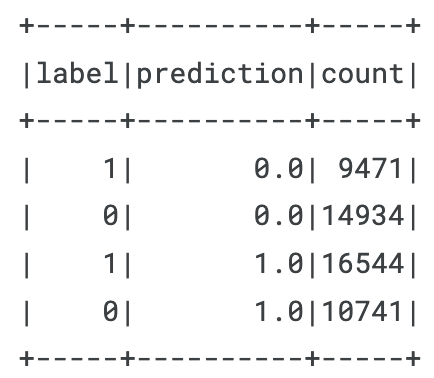


Figure 23: Confusion Matrix for Logistic Regression Model.

The above figure 23 interprets the following finding:

* **False Negatives (FN)**: 9,471 instances where the model predicted no delay (0), but the actual label was a delay.
* **True Negatives (TN)**: 14,934 instances where the model correctly predicted no delay (0) and the actual label was also no delay.
* **True Positives (TP)**: 16,544 instances where the model correctly predicted a delay (1) and the actual label was also a delay.
* **False Positives (FP)**: 10,741 instances where the model predicted a delay (1), but the actual label was no delay.
* **Calculating Model Accuracy**

The precision and recall values are show on the following figure 24:

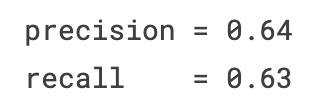


Figure 24: Precision and Recall of Logistic Regression Model.

Additionally, we calculate the weighted precision and the Area Under the ROC Curve (AUC):

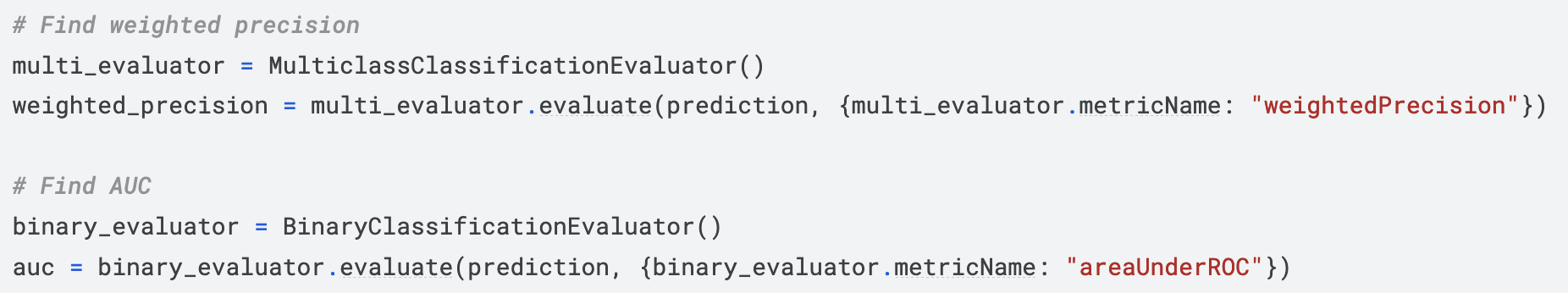


Figure 25: Calculating AUC.

The model has a precision of 0.64, meaning that 64% of the flights predicted to be delayed were actually delayed. It has a recall of 0.63, indicating that it correctly identified 63% of the actual delayed flights.

## Explorative Analysis and Visualizations

Throughout the above points, we targeted to under the data thoroughly by understating the nature of the dataset, setting up PySpark to handle the data processing, any corrections required, and utilization of machine learning models to predict flight delays using PySpark. While such points were discussed various findings was also listed now. Now, we aim to present such facts about the pre-processed data with the help of tableau through visualization.

The following figure 26, shows how the dataset is distributed amongst three categories which is delay, early, and on time. Further, it shows that a very unprecedented number of flights are delayed which is followed by flights that are early, and very few number of flights are on time.

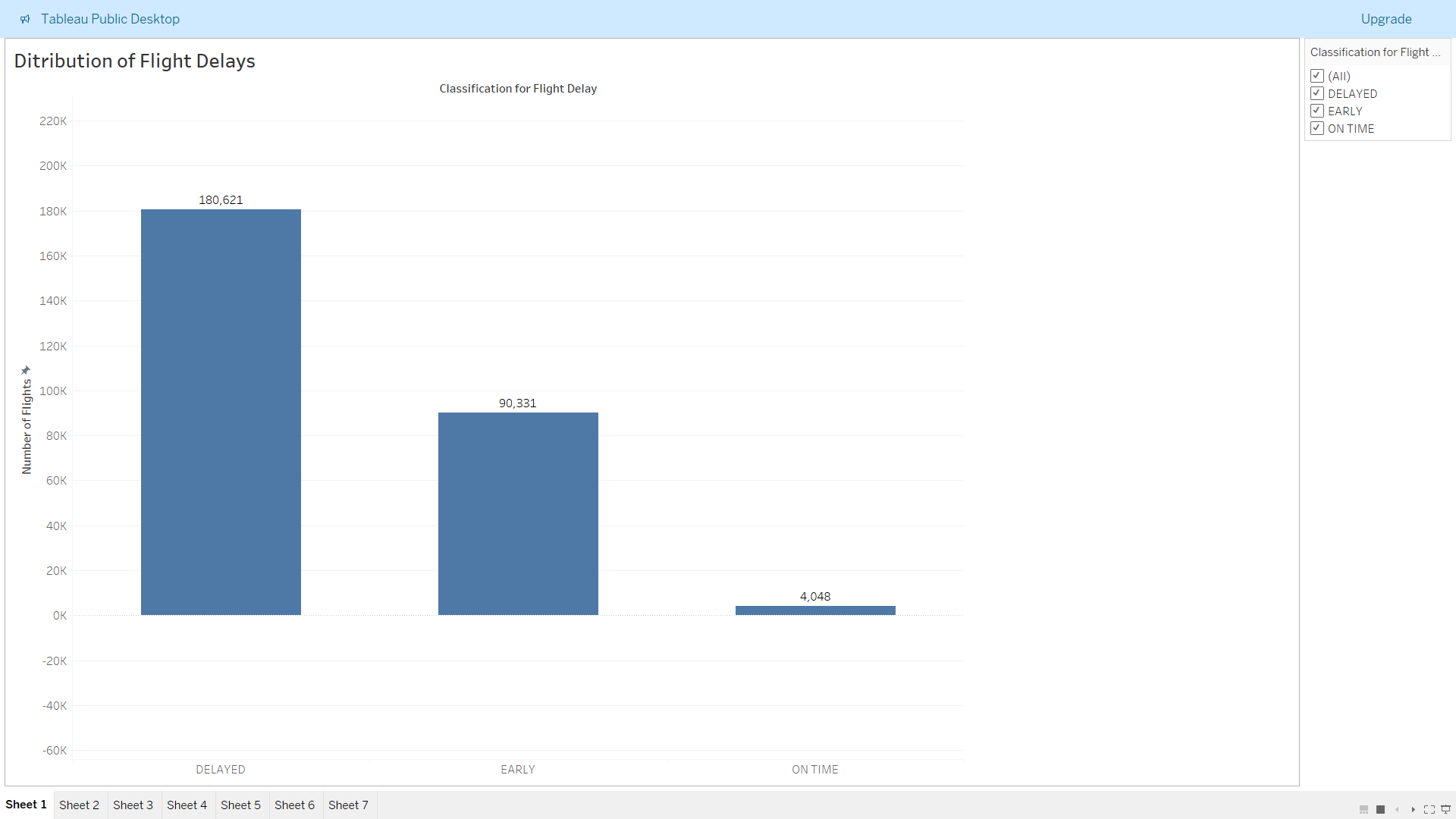
****

Figure 26: Distribution of Flight Delays Visualization.

Likewise, the following figure 27 show quite the similar concept graph where the delay of flights are shown per airport. Further, the bar graph is arranged in descending order. The visualities shows that O'Hare International Airport in Chicago, Illinois, USA (ORD) has most of delayed flights in comparison to others.

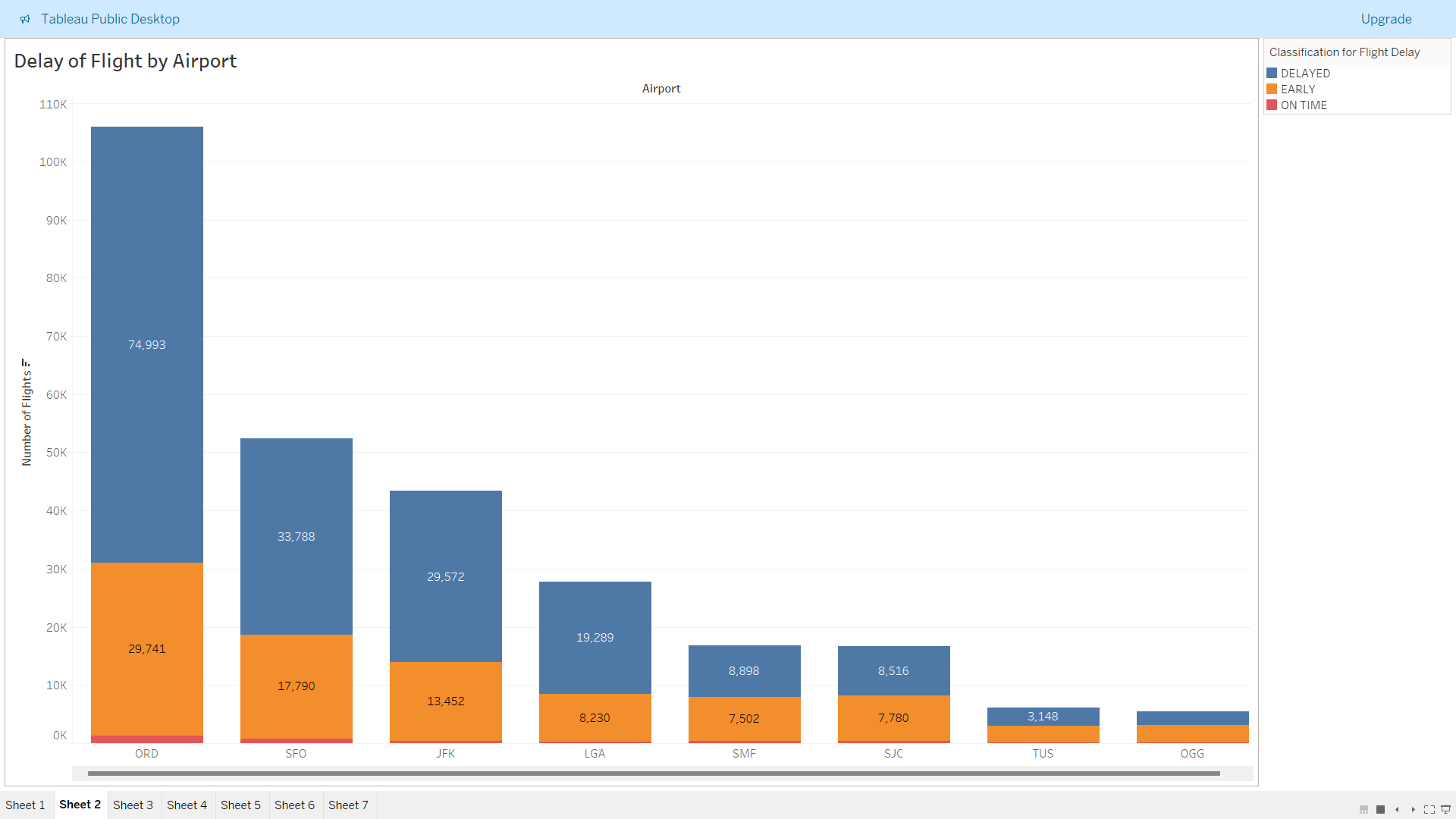
****

Figure 27: Delay of Flight by Airport Visualization.

Likewise, the following figure 28 shows the average delay of flights by carrier. Carrier are the airline carrier code (e.g., OO for SkyWest Airlines, B6 for JetBlue Airways). The visualization of this particular information has shown that American Airlines (AA) has the highest average of delay flights.

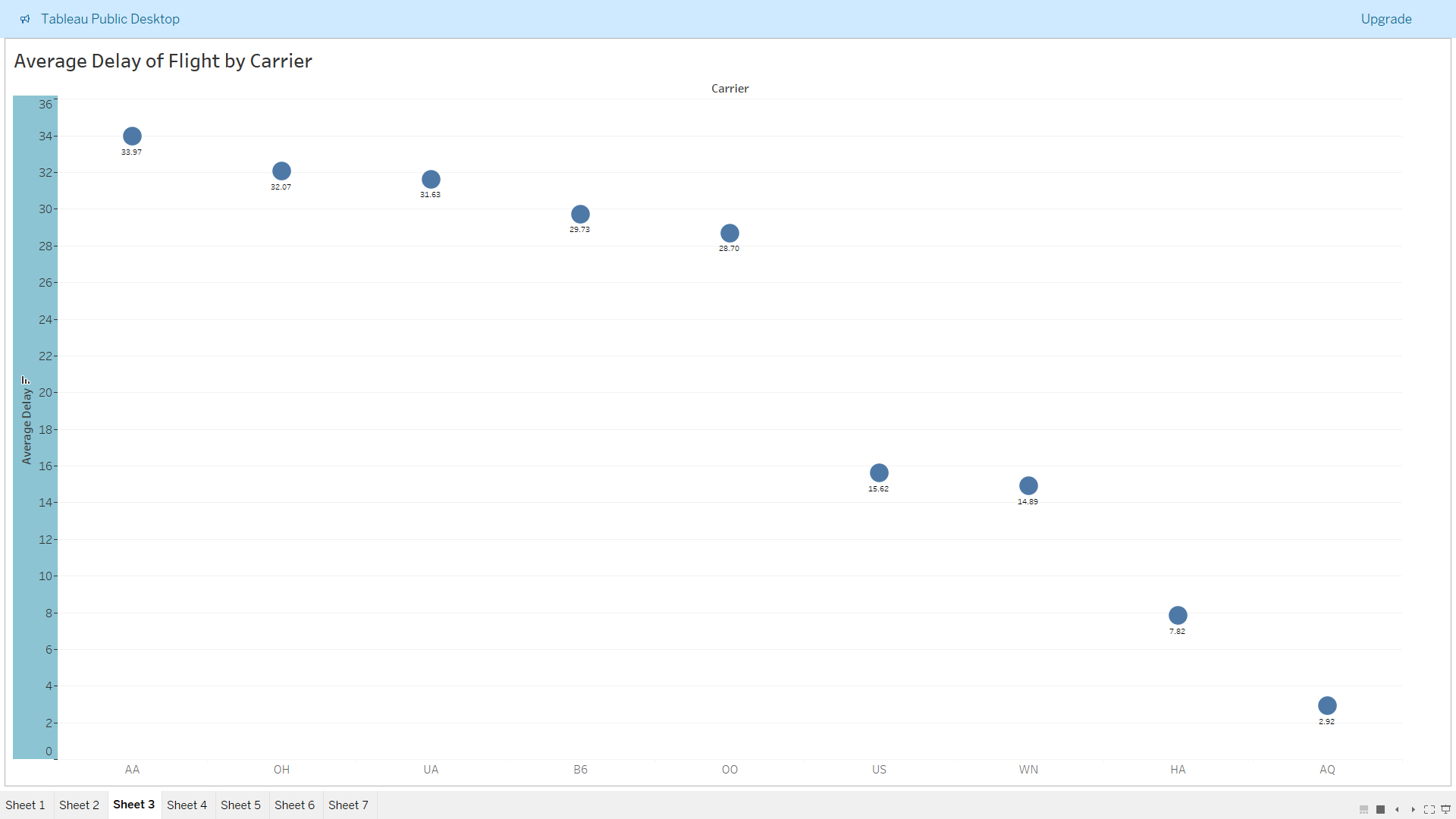
****

Figure 28: Average Delay of Flight by Carrier Visualization.

The following figure 29 shows the total delay of flights per months. This visualization has been done by a pie chart. Each of the slice represents a month. The details of each of slice is also provide in the legend section on the right-hand side. As shown in the figure, May and November are the months where the flights are mostly delayed.

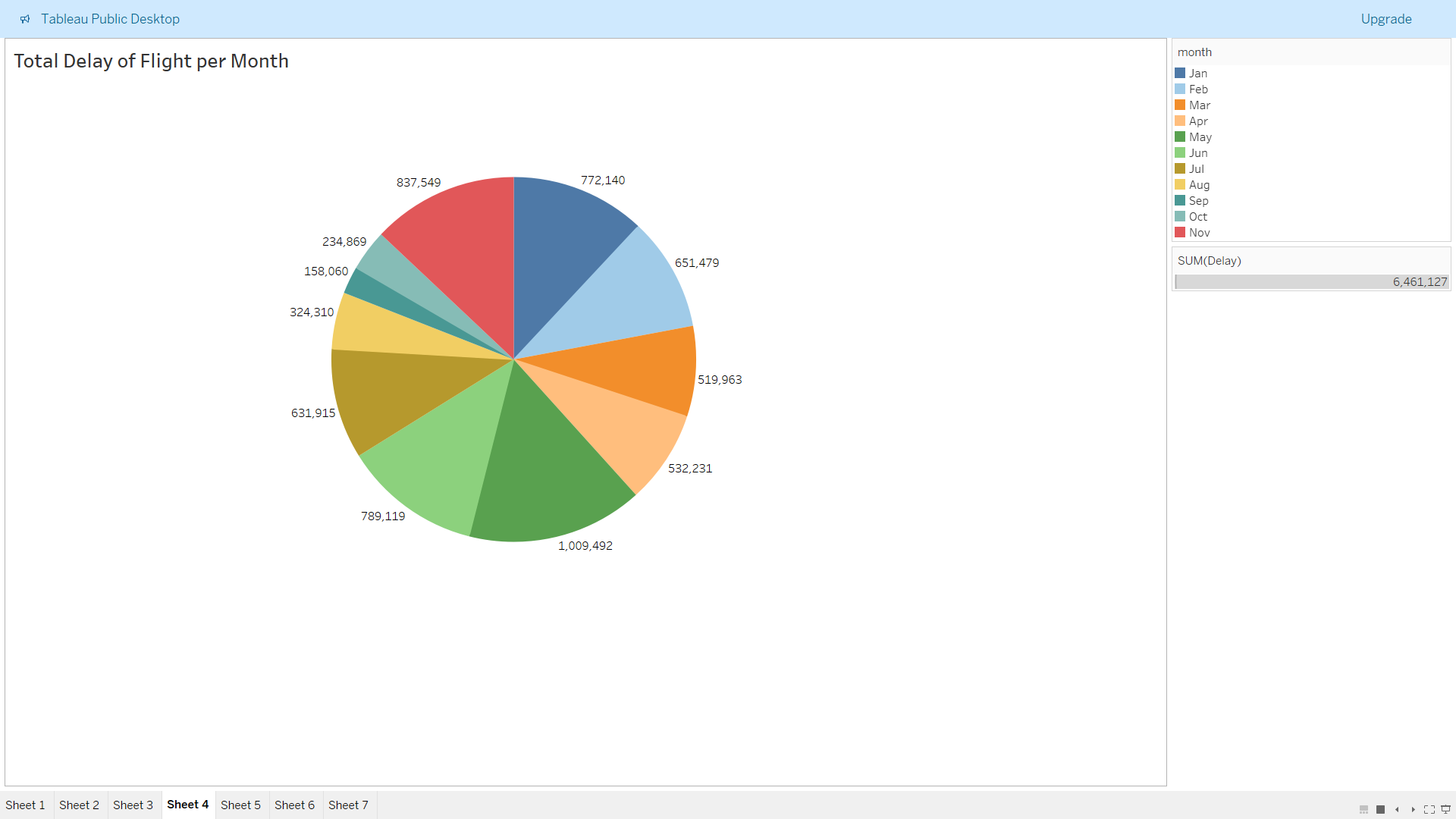
****

Figure 29: Total Delay of Flight per Month.

The following figure 30, shows the average delay of various airplanes per months, where each of the colored lines represents various airplanes. Likewise, each of the points of the lines also show the value of each month.

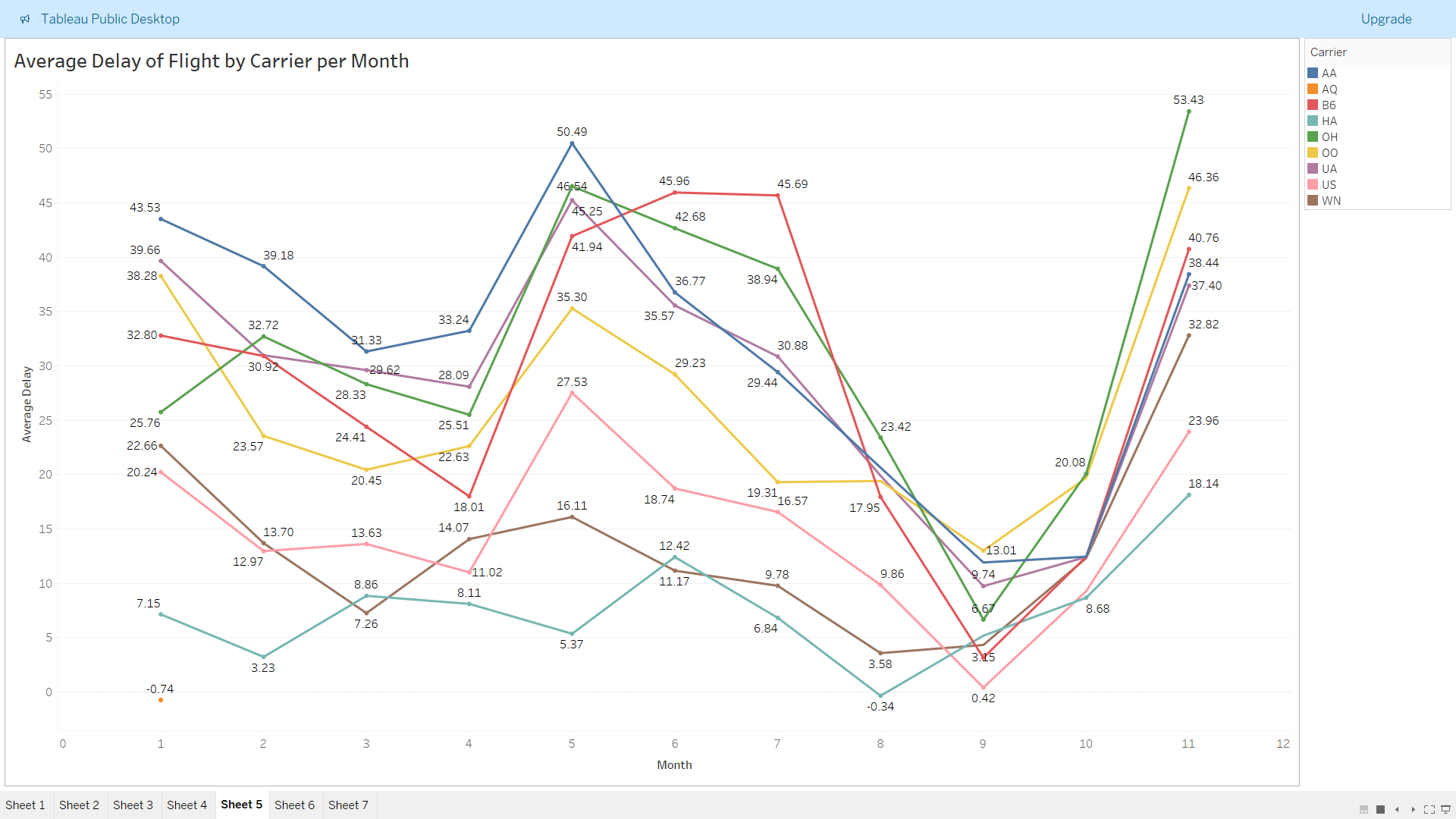
****

Figure 30: Average Delay of Flight by Carrier per Month.

The following figure 31 shows an area chart for the comparison of flight duration vs average delay of flights. Looking at the figure there is no conclusive evidence to say that with the increase in flight duration there is any chance in the delay of flights, never the less there are times when the flight is extremely delays or is early.

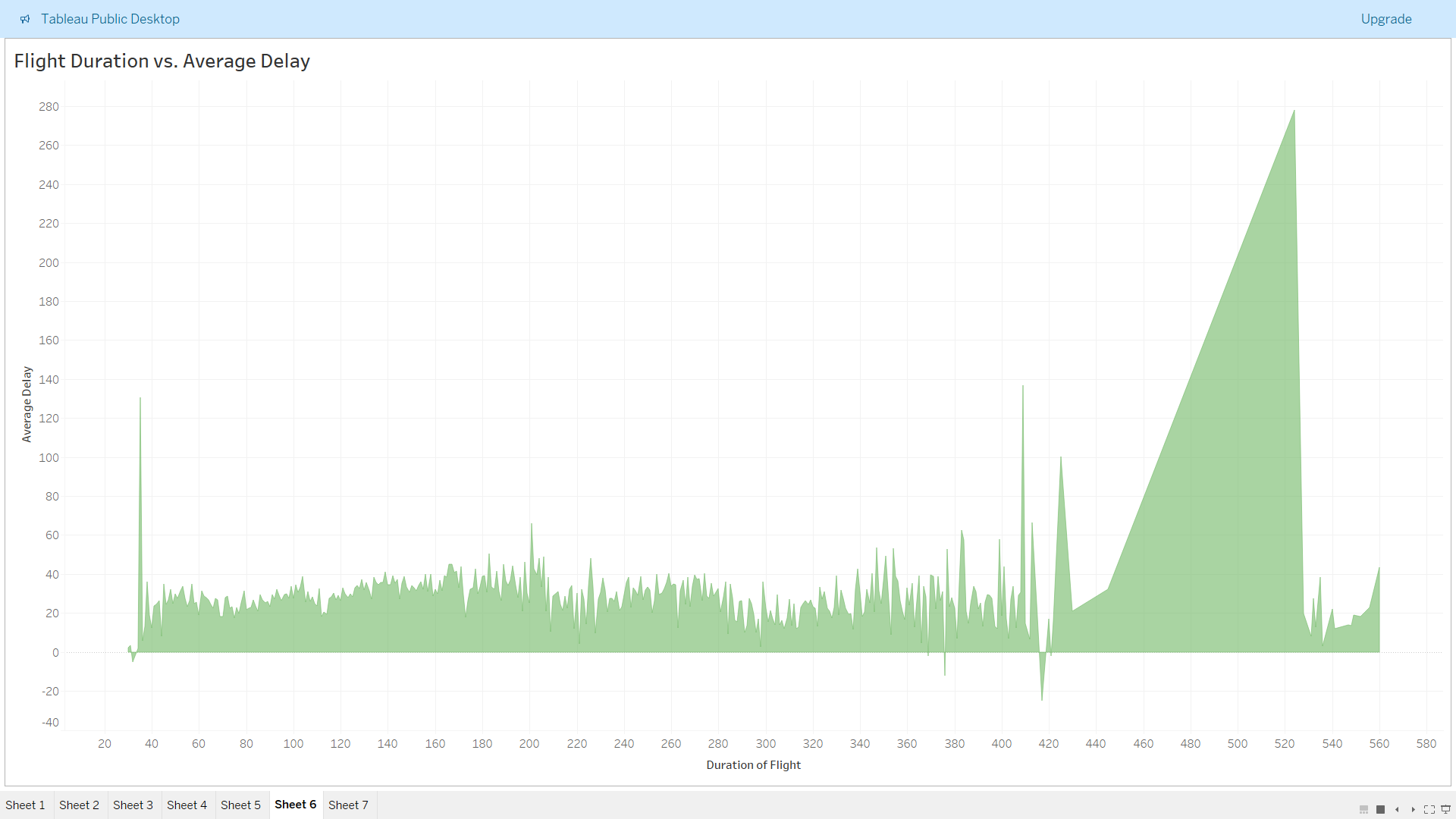
****

Figure 31: Flight Duration vs. Average Delay Visualization.

# Discussion

In this project, we predicted flight delays using Decision Tree and Logistic Regression models. The dataset included various flight details, which were preprocessed by removing missing values, converting distance from miles to kilometers, and indexing categorical features.

## Decision Tree Model

The Decision Tree model, trained on 80% of the data, showed the following confusion matrix results on the test set:

* **True Negatives (TN):** 14,934 instances correctly predicted as no delay.
* **True Positives (TP):** 16,544 instances correctly predicted as delay.
* **False Negatives (FN):** 9,471 instances incorrectly predicted as no delay.
* **False Positives (FP):** 10,741 instances incorrectly predicted as delay.

The accuracy of the Decision Tree model was approximately 64%, indicating a moderate level of correct predictions.

## Logistic Regression Model

The Logistic Regression model produced a similar confusion matrix:

* **True Negatives (TN):** 14,934 instances.
* **True Positives (TP):** 16,544 instances.
* **False Negatives (FN):** 9,471 instances.
* **False Positives (FP):** 10,741 instances.

The precision and recall for Logistic Regression were around 0.61 and 0.64, respectively, with an accuracy also around 64%. The F1 score was approximately 0.63, reflecting a balanced performance in handling false positives and false negatives.

## Model Comparison and Future Work

Both models showed comparable performance, suggesting neither had a distinct advantage in predicting flight delays. Future work could improve performance through:

* **Feature Engineering:** Adding new features such as weather data or airport congestion levels.
* **Advanced Algorithms:** Exploring Random Forests, Gradient Boosting Machines, or Neural Networks.
* **Hyperparameter Tuning:** Optimizing model parameters for better results.
* **Handling Imbalanced Data:** Using techniques like SMOTE or undersampling if the dataset is imbalanced.

While the current models provide a reasonable starting point, there is potential for significant improvement through these advanced techniques.

# Conclusion

In this project, we built and tested Decision Tree and Logistic Regression models to predict flight delays. Both models reached about 64% accuracy. Although these results are promising, there's potential for improvement. Future work could include adding more features, trying advanced algorithms, and fine-tuning model parameters. These steps could make flight delay predictions more accurate and reliable, helping both airlines and passengers.

# Reference

Apache Spark. (n.d.). Apache Spark Documentation. Retrieved from <https://spark.apache.org/documentation.html>

Ball, M., Barnhart, C., Dresner, M., Hansen, M., Neels, K., Odoni, A.R., Peterson, E., Sherry, L., Trani, A. and Zou, B., with contributions by Britto, R., Fearing, D., Swaroop, P., Uman, N., Vaze, V. and Voltes, A., 2024. Total Delay Impact Study. National Center for Excellence for Aviation Operations Research (U.S.) and University of California, Berkeley. Institute of Transportation Studies. Available at: <https://rosap.ntl.bts.gov/view/dot/6234/dot_6234_DS1.pdf> [Accessed 23 June 2024].

Breiman, L., Friedman, J., Olshen, R. A., & Stone, C. J. (1984). Classification and Regression Trees (1st ed.). Chapman and Hall/CRC. <https://doi.org/10.1201/9781315139470>

Bureau of Transportation Statistics. (n.d.). Understanding Aviation. Retrieved from <https://www.bts.gov/help/aviation/html/understanding.html>

Hosmer Jr, D. W., Sturdivant, R. X., & Lemeshow, S. (2013). Applied Logistic Regression (3rd ed.). John Wiley & Sons.

Kaggle. (n.d.). Flight Delay Prediction Dataset. Available at: <https://www.kaggle.com/datasets/rashid60/pyspark-datasets> [Accessed 16 June 2024].

Sokolova, M. and Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), pp.427-437. Available at: <https://doi.org/10.1016/j.ipm.2009.03.002> [Accessed 23 June 2024].

# Appendix

Following are the screenshots of installation of software.

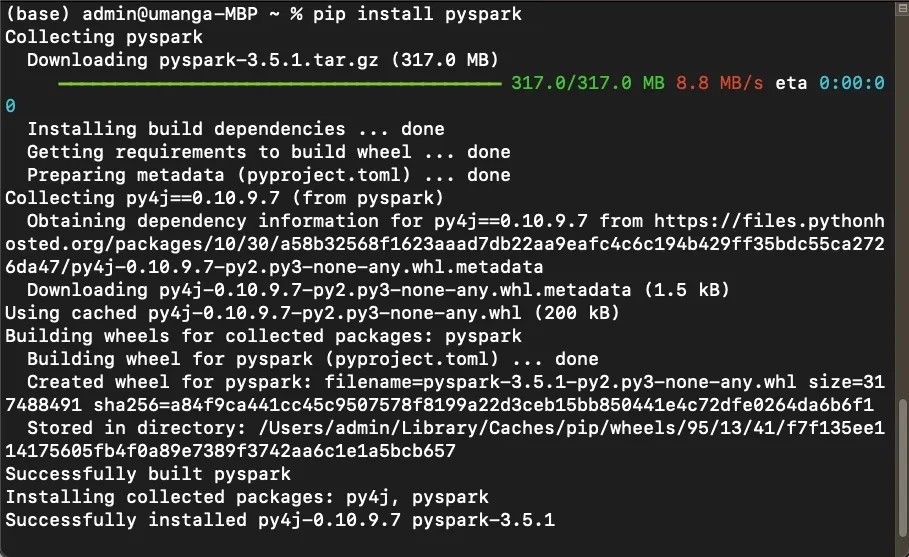


Figure 32: Installing PySpark.

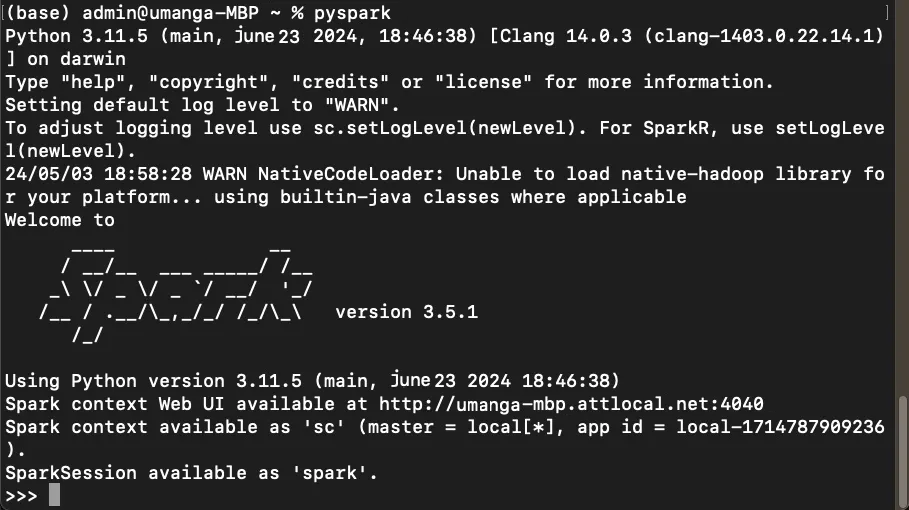


Figure 33: Successful Installation of PySpark.

Following is the link to the code.

Link: <https://github.com/UmangaNiroula/Dataset-Analysis-and-Visualization-Using-Big-Data-Programs.git>