UNIVERSITY OF CALGARY

Deep Dive Into Airline Data

by

Jubayer Ahmed, 10056991

Usman Liaqat, 10112908

Munal Akhtar, 30233125

Gopal Sharma, 30229669

A REPORT SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR COURSE – ENSF - 612

MASTER OF SOFTWARE ENGINEERING

Contents

1. Introduction and Motivation

1.1 Introduction

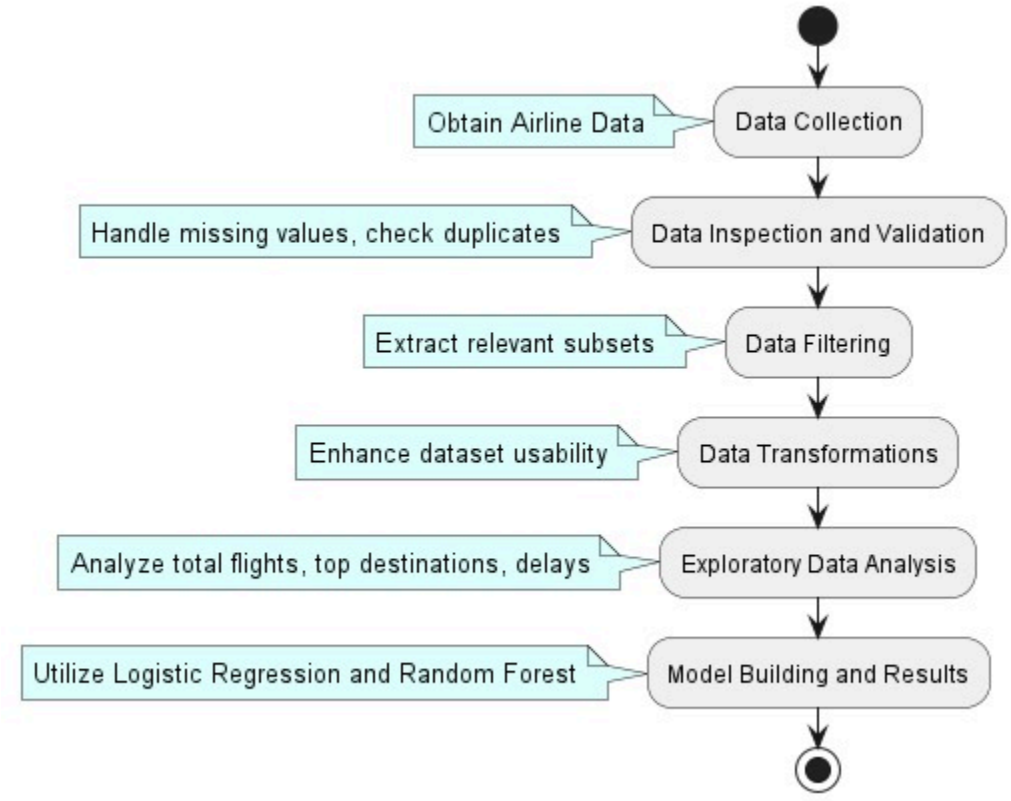
In this project we undertake a thorough analysis of a dataset comprising 120 million flight records from over 20 American airlines between the years 1987 to 2007. This comprehensive dataset carries information about flight records for major airlines operating in the USA during this period.

1.2 Motivation

Analyzing flight delays within our dataset presents a unique opportunity to extract insights valuable to stakeholders like air travelers and airlines. By uncovering the patterns and factors of flight delays, we hope to offer actionable knowledge for travelers and airlines to address systemic issues and optimize operational processes. Our study can be utilized to reduce passenger frustration, improve industry competitiveness, and contribute to a more predictable scheduling of flights.

1.3 Methodology

The diagram below provides a visual representation of the methodology applied in performing this study.



2. Data Collection

2.1 Data Sources:

For this project, we obtained the data to perform our analysis through the Harvard Dataverse via the link provided through the "Datasets for Big Data Projects" document available on D2L. We were able to download the dataset in the form of csv.bz2 files segregated by the years. The complete Airline Flight dataset consists of 120 million records. The dataset encompasses crucial information from over 20 American airlines, providing a rich and diverse set of insights into the intricacies of airline operations.

Below is the APA citation for the data set:

Harvard Dataverse. (2008). Data Expo 2009: Airline on-time data (metadata, Version 1.0) [Data set]. https://doi.org/10.7910/DVN/HG7NV7

2.2 Key Features:

2.2.1 Flight Data Size

The dataset under scrutiny comprises a vast repository of 120 million records of flight data. This sizable dataset enables us to conduct a thorough analysis, uncovering patterns, trends, and anomalies within the airline industry, specifically concerning flight delays.

2.2.2 Relevant Features

Our dataset includes many important features, offering a holistic view of airline operations. Some key features include arrival delays which are further segregated into Weather, Security, and Late Aircraft.

These features not only provide a granular understanding of the reasons behind delays but can also be used for the formulation of targeted strategies for improving overall airline efficiency and customer satisfaction.

2.2.3 Data Utilization

To optimize end-to-end run time for our program, we decided to utilize only a percentage of the data from the years 2006 and 2007. Initially, we began by considering the data for one year, but since we wanted to capture seasonality we needed to have the data for at least two years.

3. Data Inspection and Validation:

3.1 Overview

In this section, we outline the processes of inspection and validation employed to pre-process the dataset.

3.2 Data Inspection

Initially, the raw flight data comprised of the following columns:

1. Year

2. Month
3. DayofMonth
4. DayOfWeek
5. DepTime
6. CRSDepTime
7. ArrTime
8. CRSArrTime
9. UniqueCarrier
10. FlightNum
11. TailNum
12. ActualElapsedTime
13. CRSElapsedTime
14. AirTime
15. ArrDelay
16. DepDelay
17. Origin
18. Dest
19. Distance
20. TaxiIn
21. TaxiOut
22. Cancelled
23. CancellationCode
24. Diverted
25. CarrierDelay
26. WeatherDelay
27. NASDelay
28. SecurityDelay
29. LateAircraftDelay

3.2.1 Handling Missing Values

A proactive approach was taken to address missing values within the dataset to ensure a comprehensive and accurate representation of the dataset.

Since we are taking Arrival Delay to be our target vector, we dropped all the records which had the target variable as 'Null'. Subsequently, we did a null count on all of the remaining columns to ensure there were no remaining nulls in the data.

```
+----+-----+----------+---------+----------+----------+-------------+--------+------+----+--------+
|Year|Month|DayofMonth|DayOfWeek|CRSDepTime|CRSArrTime|UniqueCarrier|ArrDelay|Origin|Dest|Distance|
+----+-----+----------+---------+----------+----------+-------------+--------+------+----+--------+
|   0|    0|         0|        0|         0|         0|            0|       0|     0|   0|       0|
+----+-----+----------+---------+----------+----------+-------------+--------+------+----+--------+
```

3.3 Data Validation

3.3.1 Checking for Duplicate Records

The dataset was checked for any duplicate records to maintain the dataset's reliability.

3.3.2 Inspecting range and important statistics for numerical features

Built-in .describe function was used to compute important statistics for numeric features of the data. This encompassed measures such as count, mean, standard deviation, min, and max. The insights gained from these statistics provided a foundational understanding of the dataset's central tendencies and distributions.

summary	Year	Month	DayofMonth	DayOfWeek	CRSDepTime	CRSArrTime	Distance
count	146571	146571	146571	146571	146571	146571	146571
mean	2006.5087295576889	6.532247170313363	15.704880228694625	3.9428263435468134	1330.7448676750516	1494.814151503367	724.3678080930061
stddev	0.49992549442117584	3.427151334121104	8.779401445421444	1.9925988539597128	465.02066920726793	480.66985712367466	569.2292815711951
min	2006	1	1	1	1	1	21
max	2007	12	31	7	2359	2400	4962

4. Data Filtering

4.1 Purpose

Data filtering is pivotal for extracting subsets of data based on select features or feature values. This focused approach enables us to streamline our dataset, concentrating on specific aspects relevant to our research questions.

4.2 Key Steps in Data Filtering

4.2.1 Removal of Flights with No Takeoff

Flights that never took off, leading to "NA" values, were removed from the dataset. This step ensured the exclusion of incomplete or irrelevant records, enhancing the accuracy of subsequent analyses.

4.2.2 Focused Column Selection

To narrow the scope of our analysis, we carefully selected a subset of columns, discarding extraneous information. Columns such as 'Year,' 'DepTime,' 'ArrTime,' and others were excluded, ensuring that our analysis remains concentrated on the most pertinent features. The following list was used to remove unnecessary columns.

columns_to_remove = [

'DepTime', 'ArrTime', 'FlightNum', 'TailNum', 'ActualElapsedTime', 'CRSElapsedTime', 'AirTime', 'TaxiIn', 'TaxiOut', 'CancellationCode', 'Diverted', 'CarrierDelay', 'WeatherDelay', 'NASDelay', 'SecurityDelay', 'LateAircraftDelay', 'Cancelled', 'DepDelay'

]

5. Exploratory Data Analysis

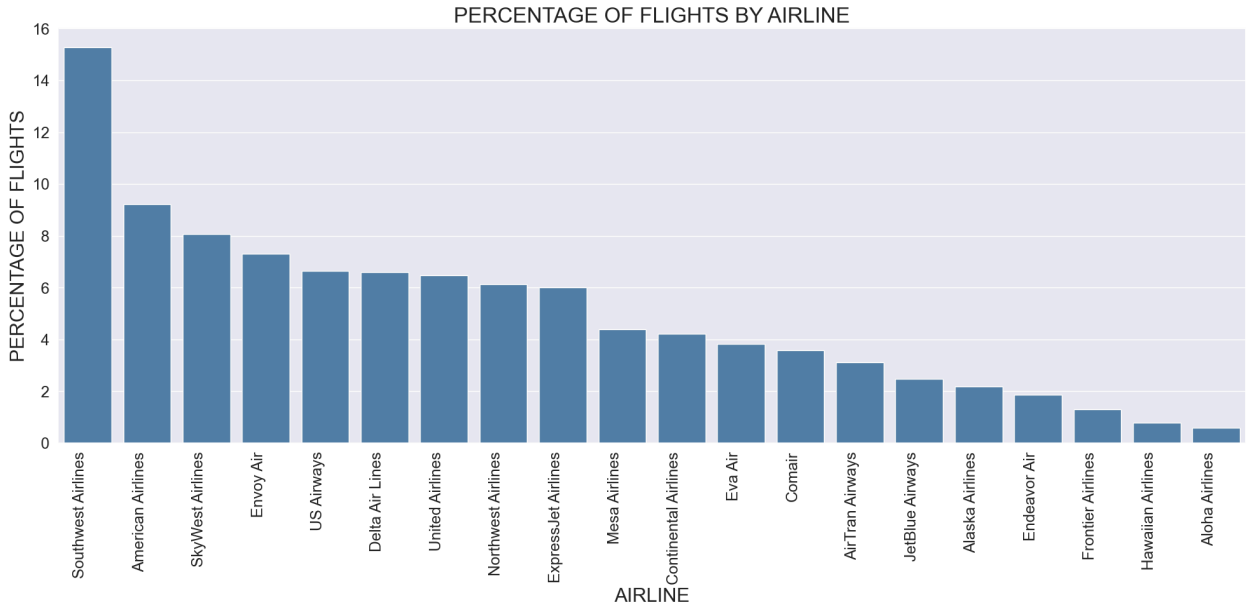
5.1 Overview

Exploratory Data Analysis (EDA) is a critical phase in our project, "Deep Dive Into Airline Data." This section delves into the insights derived from our visualizations and analyses, providing a comprehensive understanding of the delay characteristics..

5.2 Total Number of Flights by Airline

5.2.1 Key Findings

The bar chart below depicts the percentage of flights by airline and reveals significant variations in the volume of flights among carriers.



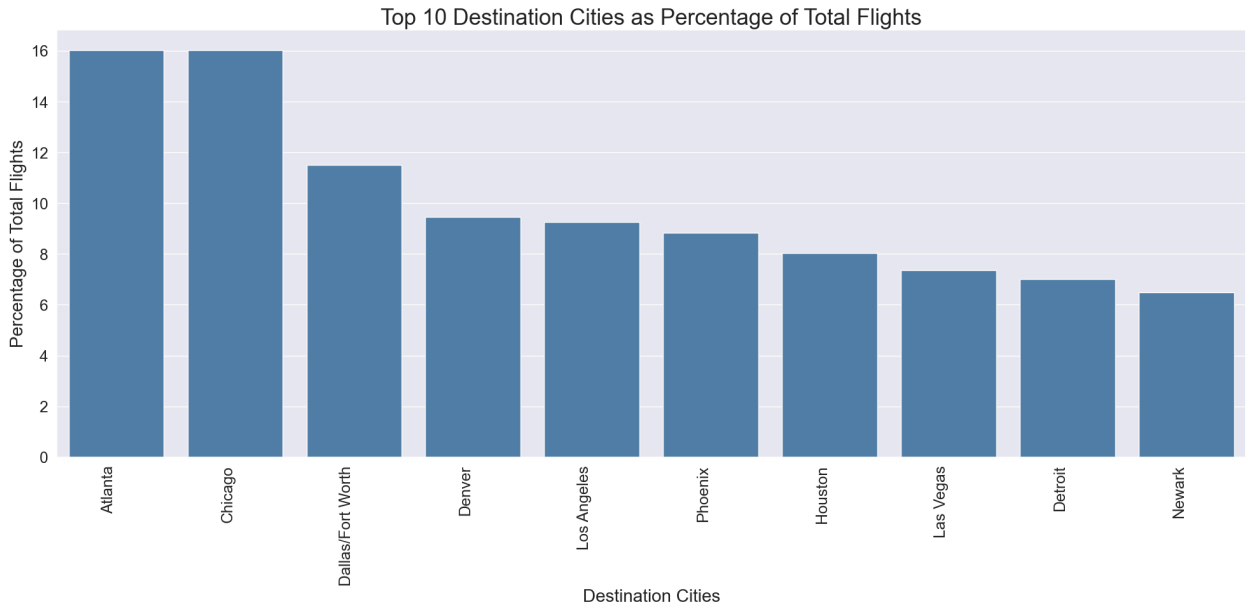
5.2.2 Observations

The top five airlines by number of flights are Southwest Airlines, American Airlines, SkyWest Airlines, Envoy Air, and United Airlines in the years 2006 and 2007 combined. These five airlines account for over half of the total number of flights in the year. The bottom five airlines by number of flights are Alaska Airlines, Endeavor Air, Frontier Airlines, Hawaiian Airlines, and Aloha Airlines. These five airlines account for approximately 5% of the total number of flights. There is a large disparity in the number of flights between the largest and smallest airlines.

5.3 Top 10 Destination Cities

5.3.1 Key Findings

The bar chart illustrating the top destination cities offers insights into the popularity of specific locations.



5.3.2 Implications

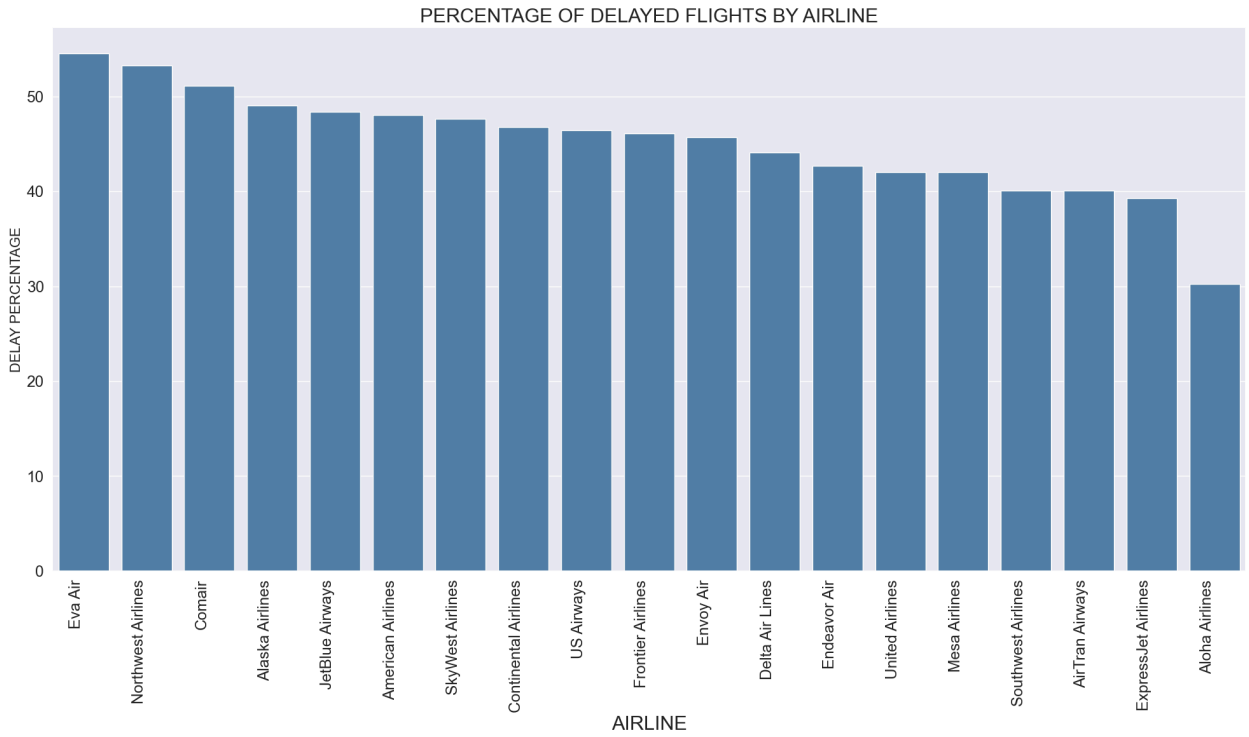
Identifying the most frequently visited destination cities informs strategic planning for routes and services.

Atlanta has the highest number of incoming flights, followed by Chicago and Dallas/Fort Worth. Although these cities have a significant lead, Denver and Los Angeles also receive a considerable volume of flights. Phoenix, Houston, Las Vegas, Detroit, and Newark round out the list with relatively similar numbers of incoming flights.

5.4 Percentage of Delayed Flights by Airline

5.4.1 Key Findings

The bar chart below highlights the percentage of delayed flights provides insights into the punctuality performance of each airline.



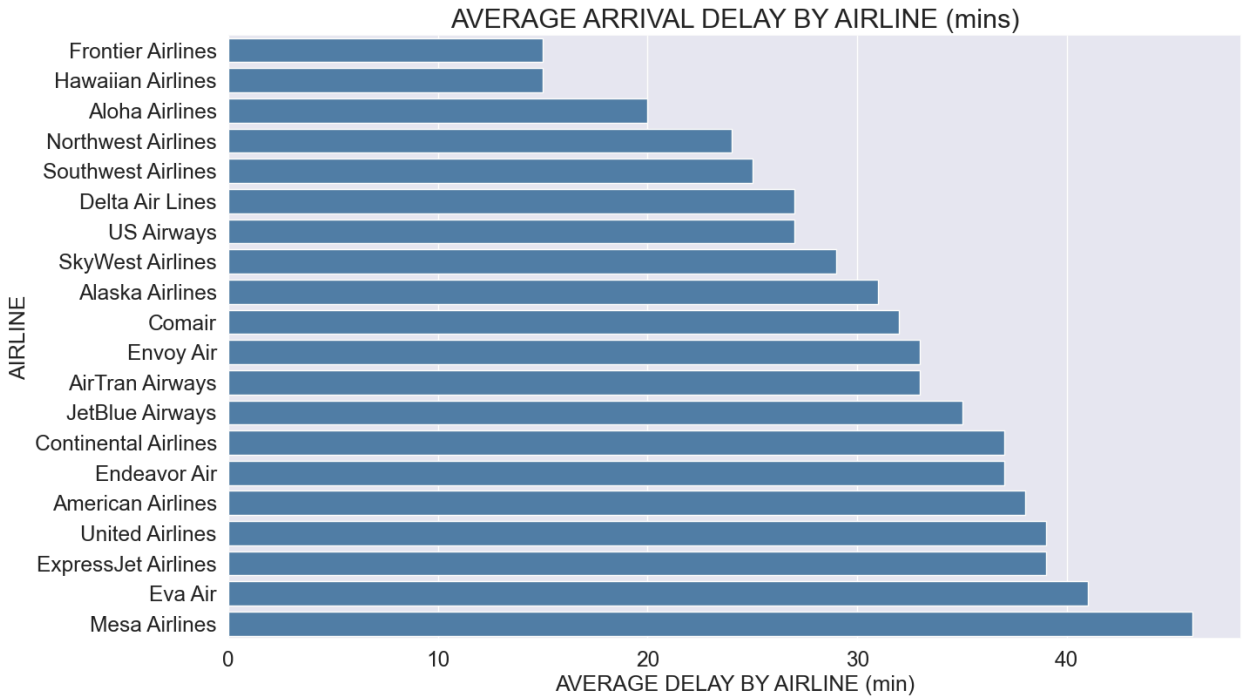
5.4.2 Observations

The chart illustrates the percentage of delayed flights by various airlines. Eva Air has the highest delay percentage at 55%, followed closely by Northwestern Airlines, at around 53%. Comair has a delay percentage of 51%. The airline with the lowest delay percentage is Aloha Airlines at 28%. The performance of Aloha Airlines is much better compared to its peers.

5.5 Average Arrival Delay by Airline

5.5.1 Key Findings

The bar plot depicting average arrival delays in minutes across various airlines showcases variations in punctuality.



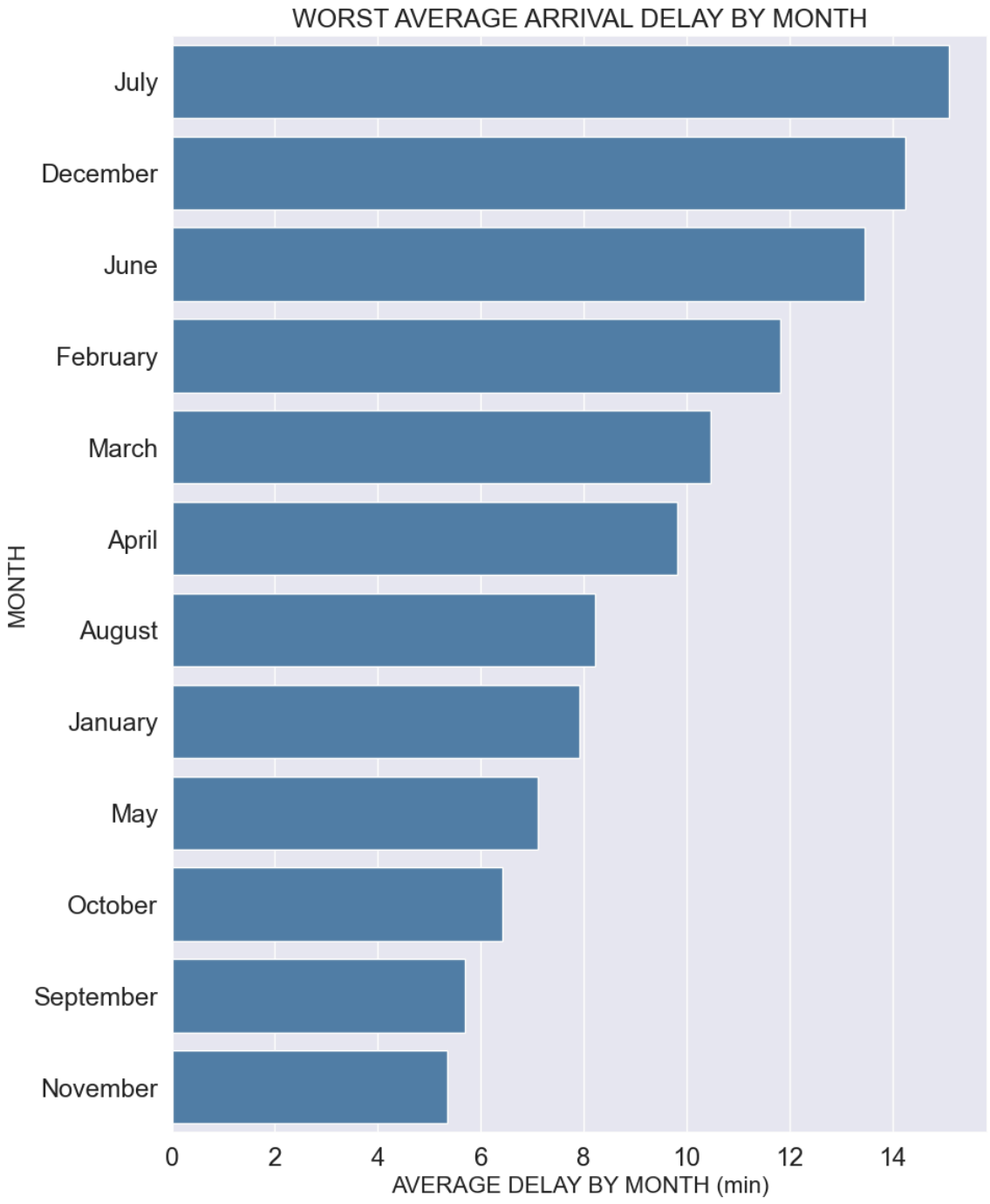
5.5.2 Observations

Mesa Airlines has the highest average delay of approximately 47 minutes among all the airlines shown in the graph. This indicates that Mesa Airlines is the least punctual airline in terms of arrival time. Frontier Airlines and Hawaiian Airlines have the lowest average delay of around 15 minutes each. This suggests that these two airlines are the most reliable in terms of arrival time. There is a large variation in the average delay among different airlines, ranging from roughly 15 minutes to more than 40 minutes. This implies that the quality of service and the efficiency of operations vary significantly across different airlines.

5.6 Worst Months for Delays

5.6.1 Key Findings

The bar plot depicting the worst average arrival delay by month across airlines showcases variations in punctuality.



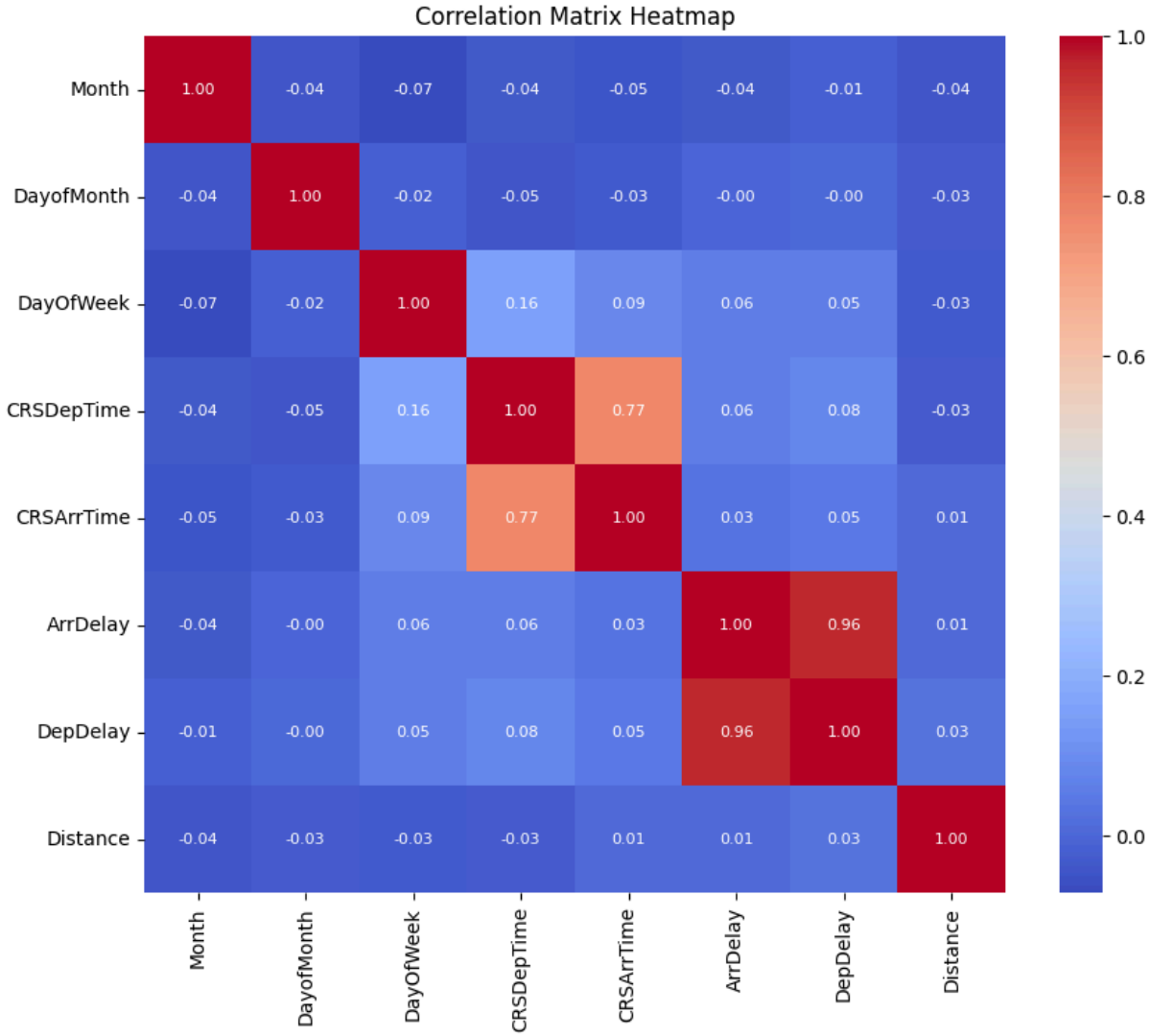
5.6.2 Observations

The months with the worst average arrival delays are July, December, June, February, March, April, August, January, May, October, September, and November. The months with the best average arrival delays are October, September, and November. There is a large range in the average arrival delay between the months, from about 4.5 minutes in November to about 16 minutes in July. The reasons for the differences in arrival delays between the months are likely

complex and varied but could include factors such as weather patterns, holiday travel, and air traffic congestion.

5.6.3 Correlation Matrix

Insights derived from a correlation matrix are particularly valuable in understanding the interdependencies between features within the dataset. They can guide subsequent analyses and decision-making processes.



By analyzing the correlation matrix, we find that most of the features are not correlated to the arrival delay except for the departure delay feature, which has correlation coefficient of 0.96 with

arrival delay. However, a passenger would want to know if his flight going to be delayed before he arrives at the airport or before they enter the plane to avoid a delayed flight therefore, the departure delay feature cannot render itself as useful for a machine learning model which predicts flight delay, therefore we will drop the departure delay variable from the feature matrix.

6. Data Transformations

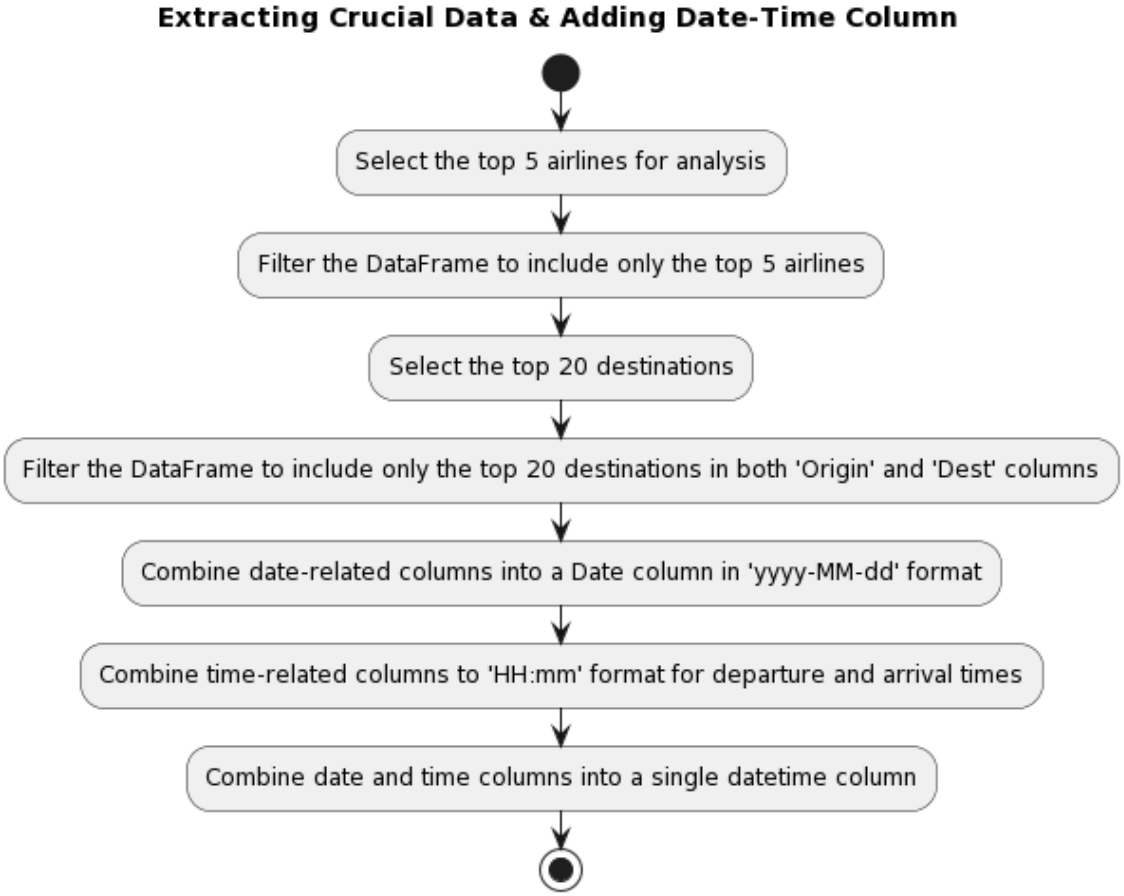
6.1 Purpose

Data transformations involve manipulating the dataset to enhance its usability and align it with the requirements of our analytical methodologies. This section outlines the key transformations applied to our airline dataset.

6.2 Transformation Steps

6.2.1 Extracting crucial data and adding Date-Time column to capture the temporal component of data

In line with the steps shown in the following flow chart, we extracted crucial data which is a subset of the original data, this subset pertains to the top 5 airlines and top-20 destinations. Furthermore, we added a Date-Time column to capture the temporal component of the data.



6.2.2 Classifying Delays

Subsequently, we decided to label the delays. This was done by adding a new column ‘label’ which was created by applying conditions and labels to the ‘ArrDelay’ column (our target vector). The conditions are defined as follows:

- If the ‘ArrDelay’ value is less than or equal to 10, the label is 1.
- If the ‘ArrDelay’ value is greater than 10 and less than or equal to 60, the label is 2.
- If the ‘ArrDelay’ value is greater than 60 and less than or equal to 120, the label is 3.
- For all other cases, the label is 4.

The original ‘ArrDelay’ column is consequently dropped from the DataFrame.

6.2.3 Data Scaling, Encoding & Pre-processing Pipeline

Before feeding data into machine learning models, scaling functions were employed. This step ensured that all features were on a comparable scale, preventing the dominance of certain features in the modeling process.

Categorical features such as 'Origin,' 'Destination,' and 'Carrier' were subjected to one-hot encoding. This transformation facilitated the integration of categorical variables into our analytical models effectively.

To pre-process the data systematically, a pipeline was created which consisted of scaling, encoding, and vector assembly. In particular, we followed the following steps.

1. The data was split into training and test sets while maintaining temporal order by ordering based on the date-time column.
2. Numeric and categorical features were defined.
3. Numerical features were assembled into a single vector using `VectorAssembler`.
4. Numerical features were scaled using `StandardScaler`.
5. Categorical features were indexed and encoded using `StringIndexer` and `OneHotEncoder`.
6. All features were combined into a single vector using `VectorAssembler`.
7. Finally, a pipeline for preprocessing was created using all the above stages.

The resulting pipeline was used to preprocess data before feeding it into a machine learning model.

7. Model Building and Results

7.1 Logistic Regression

7.1.1 Logistic Regression- Model Creation and Utilization

Since we decided to go with classification, logistic regression was our first choice for a machine learning model. Logistic Regression is a simple and interpretable multi-label classification model while also being fast in training and prediction, which made it our choice. The steps we followed for utilizing the logistic regression model are detailed below for quick reference:

1. A logistic regression model was created for multiclass classification.
2. A pipeline was initiated with the preprocessor and the model.
3. A ParamGrid was created for hyperparameter tuning, in this case, we tuned the regParam - **regularization parameter** with three different values: 0.01, 1, and 10.
4. An evaluator was defined for multiclass classification. In this case, we used the MulticlassClassificationEvaluator with the metric name = “accuracy”..
5. Cross-validation was performed using the pipeline, ParamGrid, and evaluator.
6. The best model from cross-validation was obtained.
7. The best parameters for the model were obtained.
8. The best training score was obtained.
9. The best validation score was obtained.
10. Predictions were made on the test data using the best model.
11. Finally, the model was evaluated on the test data.

7.1.2 Logistic Regression- Results

Best Parameters:

```
aggregationDepth: 2
elasticNetParam: 0.0
family: auto
featuresCol: features
fitIntercept: True
labelCol: label
maxBlockSizeInMB: 0.0
maxIter: 100
predictionCol: prediction
probabilityCol: probability
rawPredictionCol: rawPrediction
regParam: 1.0
standardization: True
threshold: 0.5
tol: 1e-06
```

Best Training Score: 0.6618259360539512

```
Best Validation Score: 0.6622950819672131
Test Set Score: 0.6467391304347826
```

Note:
For the linear regression model, the **regParam** parameter controls the regularization strength of the model. The **maxIter** parameter controls the maximum number of iterations. The **elasticNetParam** parameter controls the mixing parameter between L1 and L2 regularization. The **tol** parameter controls the convergence tolerance for iterative algorithms. The **fitIntercept** parameter controls whether to fit an intercept term. The **standardization** parameter controls whether to standardize the training features before fitting the model. The **family** parameter specifies the error distribution to be used in the model. The **featuresCol, labelCol, predictionCol, probabilityCol,** and **rawPredictionCol** parameters specify the names of the columns used for features, labels, predictions, probabilities, and raw predictions, respectively.

7.2 Random Forest Classifier

7.2.1 Random Forest Classifier- Model Creation and Utilization

Owing to its simplicity and established effectiveness for classification scenarios, we selected the Random Forest classifier as our second choice for applying machine learning to our dataset. The steps we followed for utilizing the random forest classifier model resemble closely those followed for logistic regression, but have been mentioned nonetheless, for the sake of completeness:

1. A Random Forest Classifier was created for multiclass classification.
2. A pipeline was defined with the preprocessor and the classifier.
3. A ParamGrid was created for hyperparameter tuning. In this case, we used the addGrid method to add and configure the parameter grid in conjunction with ParamGridBuilder. The numTrees, maxDepth, and maxBins parameters of the random forest classifier were tuned with different values.

    - The **numTrees** parameter controls the number of trees in the forest.(*tuned for values: 1,3,5*)
    - The **maxDepth** parameter controls the maximum depth of each tree. (*tuned for values: 1,3,5*)
    - The **maxBins** parameter controls the maximum number of bins used for splitting features. (*tuned for values: 2,3,5*)

4. An evaluator was defined for multiclass classification. In this case, we used the MulticlassClassificationEvaluator with the metricName = “accuracy”.
5. Cross-validation was performed using the pipeline, ParamGrid, and evaluator.
6. The best model from cross-validation was obtained.
7. The best parameters for the model were obtained.
8. The best training score was obtained.
9. The best validation score was obtained.

10. Predictions were made on the test data using the best model.
11. Finally, the model was evaluated on the test data.

7.2.2 Random Forest Classifier- Results

```
Best Parameters:
bootstrap: True
cacheNodeIds: False
checkpointInterval: 10
featureSubsetStrategy: auto
featuresCol: features
impurity: gini
labelCol: label
leafCol:
maxBins: 2
maxDepth: 1
maxMemoryInMB: 256
minInfoGain: 0.0
minInstancesPerNode: 1
minWeightFractionPerNode: 0.0
numTrees: 3
predictionCol: prediction
probabilityCol: probability
rawPredictionCol: rawPrediction
seed: 4907922549250522952
subsamplingRate: 1.0


Best Training Score: 0.6622378463299177
Best Validation Score: 0.6622950819672131
Test Set Score: 0.6467391304347826
```

Note:

For the Random Forest model, the **bootstrap** parameter controls whether to use bootstrapping when building trees. The **cacheNodeIds** parameter controls whether to cache node IDs for each instance. The **checkpointInterval** parameter controls the checkpoint interval for the model. The **featureSubsetStrategy** parameter controls the number of features to consider for splits at each tree node. The **impurity** parameter specifies the impurity measure used for information gain calculation. The **labelCol** parameter specifies the name of the label column. The **maxBins** parameter controls the maximum number of bins used for splitting continuous features. The **maxDepth** parameter controls the maximum depth of each tree. The **maxMemoryInMB** parameter controls the maximum memory in MB allocated to histogram aggregation. The **minInfoGain** parameter controls the minimum information gain for a split to be considered at a tree node. The **minInstancesPerNode** parameter controls the minimum number of instances

each child must have after a split. The **minWeightFractionPerNode** parameter controls the minimum fraction of the sum of instance weights required in each child after a split. The **numTrees** parameter specifies the number of trees in the forest. The **predictionCol** parameter specifies the name of the prediction column. The **probabilityCol** parameter specifies the name of the column storing the predicted class probabilities. The **rawPredictionCol** parameter specifies the name of the column storing the raw prediction (a.k.a. confidence) scores. The **seed** parameter specifies the random seed for the algorithm. The **subsamplingRate** parameter specifies the fraction of the training data used for learning each decision tree.

7.3 Summary & Conclusion

Below is a list of key steps performed and corresponding observations made while working on the project.

1. Owing to large amounts of data at hand, we started off by working on a portion of data for the year 2007. During our EDA, as we realized the data had a temporal component to it, we decided to consider the data for the years 2006 and 2007.
2. Since the data had a number of features that did not yield any benefit to our analysis, we had to drop most of them and work on the features (numerical and categorical) which were relevant to our classification target.
3. We were interested in classifying the delays, therefore we labeled the delays and consequently used Logistic Regression Classifier and Random Forest Classifier machine learning models.
4. Utilizing scaling for numerical data and encoding for categorical data. Moreover, to capture temporal component of the data we also included Date-time column. To apply preprocessing we used pipelines, and for parameter tuning we utilized parameter grids along with cross validation.
5. We obtained training and validation scores of around 66 % for both of the models, whereas, we got around 64 % for test scores.
6. The dataset's extensive volume and diverse array of features make it well-suited for Exploratory Data Analysis (EDA) which is corroborated by the correlation matrix shown in the EDA section of this report. The correlation matrix shows most features are not correlated with the exception of very strong correlation between departure delay with arrival delay (target variable).
7. However, the machine learning model developed in this exercise performs adequately in classifying the delay given features: airline, origin, and destination.