INTRODUCTION

Flight delays are a prominent challenge in the aviation industry, disrupting schedules and causing significant inconvenience for passengers, airlines, and airports alike. Delays are not only a source of frustration for travelers but also result in substantial economic costs and operational inefficiencies for airlines and airports.

Our project seeks to address two main questions: (1) **Will a flight operated by an airline be delayed?** and (2) **If delayed, how critical is the delay?** By predicting both the likelihood of a delay and its estimated duration, this project aims to provide insight for better managing flight disruptions. Using [insert feature], to determine whether a flight will be delayed and by how long

The insights generated from this project will benefit multiple stakeholders. **Airlines** can optimize schedules, allocate resources more effectively, and proactively communicate with passengers, enhancing both efficiency and customer satisfaction. **Passengers** will benefit from reduced uncertainty and the ability to plan alternate arrangements in advance. **Airports** can use delay predictions to streamline runway and gate operations, minimizing bottlenecks and improving the overall travel experience.

By addressing these research questions, this project contributes to solving a common problem in the aviation industry and ultimately improves operational & cost efficiency, passenger satisfaction,and overall a more reliable and efficient air travel system.

DATASET

The “Flight Status Prediction” dataset on Kaggle includes flight cancellation and delay data between 2018 and 2022. We chose to use the 2021 dataset because it is the most recent dataset that covers the whole cycle of a calendar year.

Before cleaning, the 2021 dataset contains 6,311,871 rows and 61 columns. From the 61 columns, we chose 21 columns as relevant or potentially important features for our modeling objectives. We chose 3 columns as our target variables (1 for each type of prediction task – continuous, binary, and categorical). We also categorized the 21 features into 7 separate groups based on their type of information and their correlations. We plan to choose at most 1 feature from each group to avoid redundancy and multicollinearity. The exact choice will be informed by model performance and evaluation results.

The 7 feature groups include:

1. **Flight Date:** *When is the scheduled flight date?*
2. **Flight Time:** *When is the scheduled departure time or arrival time?*
3. **Airline:** *Which airline will operate the flight?*
4. **Flight Number & Aircraft Number:** *What’s the flight number & which aircraft was used?*
5. **Origin Location:** *Where is the flight planned to take off from?*
6. **Destination Location:** *Where is the flight planned to land?*
7. **Distance:** *What’s the distance between planned origin and destination airports?*

Below are our target variables for each type of prediction:

- Continuous Prediction: `ArrDelayMinutes`
- Binary Prediction: `ArrDel15`
- Categorical Prediction: `ArrivalDelayGroups`

After selecting the relevant columns, we performed data cleaning by addressing the granularity of data, data types, missing values, invalid entries, and duplicated records. For our EDA efforts, we checked summary statistics, distribution, correlation, and unique values for non-numerical variables, and detected outliers. Some key observations include class imbalance (with around 83% belonging to the negative class), skewed distributions, and a diverse set of unique values. We plan to use some proxy features to reduce the number of one-hot encoded columns. The Appendix section of the report includes plots and tables with additional details.

We conducted feature engineering to derive new variables and refine existing ones to make our dataset more informative and suited for predictive modeling. These features aimed to capture operational, temporal, and contextual factors influencing flight delays. Temporal features such as weekends and seasons were created to reflect travel patterns and seasonal weather impacts. For instance, Season was one-hot encoded into categories like Season_spring and Season_winter to capture weather-driven trends.

We also introduced operational features, such as Origin_capacity and Dest_capacity, which quantified airport congestion by calculating the percentage of total flights originating from or arriving at specific airports. Additionally, AvgDelaysPerAircraft was derived to reflect patterns in aircraft-specific delay frequencies, helping us identify operational inefficiencies like maintenance issues. Interaction features like Distance_AvgDelays were engineered to explore the relationship between flight distance and historical delays.

Finally, categorical variables such as Operating Airlines were one-hot encoded to ensure compatibility with machine learning algorithms.

METHODS

As our project focuses on predicting flight delays, we explored (1) whether or not a flight would be delayed, and (2) analyzed the severity of the delays in terms of the amount of time lapsed. To answer these questions, we first considered ***who*** would want this information and then discussed ***what*** types of output would be most relevant. After brainstorming various model approaches to the problem, we divided our methodology into three iterative phases.

Cleaning/Pre-processing/EDA
To prepare the dataset for modeling, we removed missing values, engineered relevant features, and married the focus of the dataset to target specific attributes that would enhance output performance. After cleaning, we analyzed the dataset to uncover patterns and trends in flight delays. Through visualizations and statistical summaries, we aimed to provide insights into delay trends, helping to inform modeling decisions and guide the interpretation of results.

Modeling
The modeling phase involved building baseline models to begin our two flight delay prediction trajectories.

(1) To determine whether a flight would experience a delay, we explored multiple classification techniques, including logistic regression, decision trees, and ensemble methods. [CITATION] These methods were chosen to compare their predictive power and interpretability in identifying delayed flights within the dataset. The problem was framed as a binary classification task, where the target variable indicated whether a flight was delayed (yes or no). While we used accuracy as one measurement, the dataset was imbalanced so we shifted our metrics to prioritize F1 scores, paying extra attention to precision (TP) and recall (FP).

(2) To better understand the severity of flight delays, we employed a linear regression model to predict the number of minutes a given flight would be late. [CITATION] This approach allowed us to quantify the relationship between various predictor variables (e.g., departure time, weather conditions, airline, and distance) and the extent of delay. By focusing on the prediction of delay in minutes, we aimed to uncover patterns and factors that significantly contribute to longer delays, offering actionable insights for stakeholders in the aviation industry.

To evaluate the performance of the regression model, we used the Root Mean Squared Error (RMSE) as the primary metric. RMSE measures the average magnitude of error between the predicted delay times and the actual observed values, with lower RMSE values indicating better model performance. This metric was chosen for its sensitivity to larger errors, which aligns with our goal of accurately predicting significant delays.

Evaluation & Tuning
Our evaluation and tuning phase involved selecting relevant performance metrics, comparing models, and optimizing the outputs of each model. [1].

Through metric balancing, hypermeter tuning, cross-validation, visualization, and iterative analysis, we then refined the baseline models from the previous architecture phase.

INDIVIDUAL WORK

[Lei Zhang]: Data Cleaning, EDA and Relevant Feature Selection

For this project, I focused on data cleaning, exploratory data analysis, and selecting the relevant features and the target variables for each type of modeling. Based on my analysis, I provided recommendations and guidelines to subsequent feature engineering and modeling tasks.

Dataset Selection

The “Flight Status Prediction” dataset on Kaggle includes flight cancellation and delay data between 2018 and 2022.

I started with examining the most recent 2022 data, because I wanted our models to capture the **latest post-COVID trends** as much as possible and to improve the **real-world usefulness** of our model for upcoming years. However, the 2022 dataset only contains data between January and July 2022. Therefore, I switched to using the 2021 dataset that does cover the **whole cycle of a calendar year**, which is preferred.

Before cleaning, the 2021 dataset contains 6,311,871 rows, 61 columns.

Relevant Features Selection

I selected 21 out of the 61 columns as relevant or potentially important features or our modeling objective. I extracted hour and minute from scheduled flight times. I further categorized them into 7 different feature groups based on their **type of information** and **correlation**. The rationale is that we should avoid choosing more than 1 feature from each group during modeling to **avoid redundancy** and **multicollinearity**.

The 7 feature groups include:

1. **Flight Date:** *When is the scheduled flight date?*
2. **Flight Time:** *When is the scheduled departure time or arrival time?*
3. **Airline:** *Which airline will operate the flight?*
4. **Flight Number & Aircraft Number:** *What’s the flight number & which aircraft was used?*
5. **Origin Location:** *Where should the flight take off from?*
6. **Destination Location:** *Where should the flight land?*
7. **Distance:** *What’s the distance between planned origin and destination airports?*

I included one more convenience variable ‘Flight Date’ for easier interpretation/visualization of our data and modeling results.

Target Variable Selection

The dataset includes both departure delay and arrival delay target variables.
I made the assumption that as **passengers**, we probably care more about arrival delay than departure delay because **we care most about arriving at our destination on time**. Thus, I selected the following target variables:

- Continuous Prediction: `ArrDelayMinutes`
- Binary Prediction: `ArrDel15`
- Categorical Prediction: `ArrivalDelayGroups`

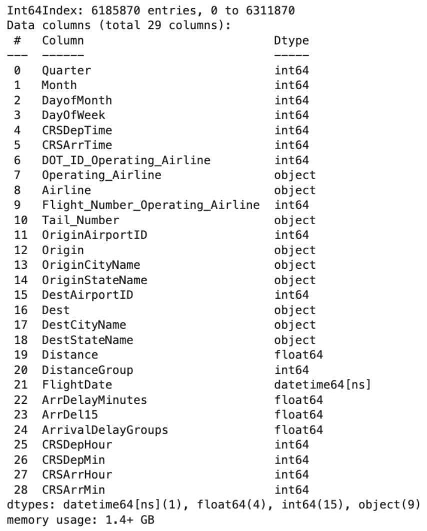
Data Cleaning

For data cleaning, I checked and cleaned the following things:

- Granularity of data

- Unnecessary columns
- Data types
- Missing values
- Invalid entries
- Duplicated records

After cleaning, the dataset contains 6,185,870 rows, 26 potentially important feature columns across 7 feature groups, and 3 target variable columns (1 target for each type of modeling task).



```
Int64Index: 6185870 entries, 0 to 6311870
Data columns (total 29 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Quarter                         int64
 1   Month                           int64
 2   DayofMonth                      int64
 3   DayOfWeek                       int64
 4   CRSDepTime                      int64
 5   CRSArrTime                      int64
 6   DOT_ID_Operating_Airline        int64
 7   Operating_Airline               object
 8   Airline                         object
 9   Flight_Number_Operating_Airline  int64
 10  Tail_Number                     object
 11  OriginAirportID                 int64
 12  Origin                          object
 13  OriginCityName                  object
 14  OriginStateName                 object
 15  DestAirportID                   int64
 16  Dest                            object
 17  DestCityName                    object
 18  DestStateName                   object
 19  Distance                        float64
 20  DistanceGroup                   int64
 21  FlightDate                      datetime64[ns]
 22  ArrDelayMinutes                 float64
 23  ArrDel15                        float64
 24  ArrivalDelayGroups              float64
 25  CRSDepHour                      int64
 26  CRSDepMin                       int64
 27  CRSArrHour                      int64
 28  CRSArrMin                       int64
dtypes: datetime64[ns](1), float64(4), int64(15), object(9)
memory usage: 1.4+ GB
```

Exploratory Data Analysis

For EDA, I checked the following things and created tables and visualizations. Visualizations can be found in the Appendix section.

- Summary statistics
- Distribution
- Outliers
- Correlation
- Unique values

Observations and Recommendations

- **Class Imbalance:** Around 82.7% of instances belong to the negative class.
- **Skewed Distributions:** Right skewed distribution of arrival delay group, arrival delay minutes, and flight distance.
- **Correlation:** Since the features within one feature group provide similar information, they have strong correlations. Consider including at most 1 feature per group.
- **Diverse Set of Unique Values:** Non-numerical features include a diverse set of unique values. For example, the airline name has 22 unique values (the least), and the origin airport abbreviation has 380 unique values.
- **Consider Using Proxy Features:** Given the diverse set of unique values, I recommended that we can use some proxy features to capture the key underlying concept. It probably makes sense to keep the airline names as one-hot. For other features, using proxies can reduce the number of one-hot encoded columns.
- **Other EDA Findings:** Summer has the highest delay rate. Sunday and Monday have the highest delay rates, but other days of week show similar delay rates. 19:00 and 3:00 scheduled departure flights have the highest delay rates. Allegiant Air has the highest delay rate, followed by JetBlue Airways.

[Audrey Folaranmi]: Feature Engineering and Baseline Model Selection Feature Engineering

For this project, I focused on creating meaningful features and removing redundant and misleading ones to improve model accuracy and generalizability. Below are the key steps I implemented:

Removing Redundant Features & Data Leakage Prevention:

- I excluded departure delay-related features (DepDelayMinutes, DepDel15, etc.) because they were strongly correlated with arrival delay metrics, which could have caused data leakage.
- I removed features like ArrDelayMinutes and ArrivalDelayGroups that directly captured the target variable, as they led to falsely high accuracy (98-100%).

Derived Features:

- Weekdays vs. Weekends: I created a categorical variable to distinguish weekends from weekdays, enabling the model to focus on broader travel patterns instead of individual day effects. This also reduced one-hot encoding complexity.
- Seasonal Trends: Using the Month column, I assigned seasonal labels (e.g., Winter, Spring) to capture weather-related effects like winter storms or summer air traffic delays.

Aggregated Features:

- Aircraft-Specific Trends: I calculated the average number of delays per aircraft using the Tail_Number column. This helped identify aircraft prone to delays, potentially signaling maintenance or staffing issues.
- I calculated the percentage of total flights per airport (origin and destination) to capture operational load, which served as a proxy for airport capacity.

Scaling and Encoding:

- The numerical features were standardized for Logistic Regression, while Decision Trees and Random Forests used unscaled data.
- For categorical variables, I tested both one-hot encoding and label encoding to balance model performance with dimensionality concerns.

Baseline Model Selection

I evaluated multiple baseline models to establish performance benchmarks and identify strengths and weaknesses for later tuning:

The majority class baseline was calculated by always predicting non-delayed flights, which provided a naive accuracy benchmark. This helped evaluate the added value of machine learning models beyond simple predictions.

For Model A, I used Logistic Regression with standardized features to ensure effective convergence and interpretability. This model served as a simple yet effective baseline for understanding the relationship between predictors and arrival delays.

For Model B, I implemented a Decision Tree Classifier. This model was selected for its ability to handle categorical variables without extensive preprocessing and capture non-linear patterns in the data. Feature importance analysis revealed key predictors, such as operational load and time-specific trends. However, the model's performance on imbalanced data indicated room for improvement, particularly in recall for delayed flights.

Model C, the Random Forest Classifier, improved generalization and reduced overfitting by combining multiple decision trees. I used feature importance from this model to guide further feature engineering and refinement.

Observations and Recommendations
The target variable, ArrDel15, was highly imbalanced (~83% non-delayed flights), so I focused on metrics like recall, precision, and F1-score rather than accuracy. Some features, particularly those related to operational delays and seasonal trends, showed promise for improving model performance. I recommended further testing and refinement of these features during the evaluation and tuning phase to enhance predictive accuracy. Finally, I noted that pre-departure features should remain the focus to align predictions with real-world use cases.

[Linda Mutesi]: Developing Linear Regression Model and Improvement

My primary responsibility was to develop and improve a linear regression model for predicting flight delay durations (ArrDelayMinutes).

Model Development

I started by defining the feature set and target variable. To ensure the model did not learn from features directly related to the target (to prevent data leakage), I excluded columns such as DepDelayMinutes, ArrDel15, and ArrivalDelayGroups. This was to ensure the model's predictions were based on independent variables and not on features related to the target.

Next, I included an interaction feature, Distance_AvgDelays, by multiplying Distance and AvgDelaysPerAircraft. This feature was added to capture the relationship between flight length and the historical reliability of aircraft, as linear regression cannot inherently model interactions. By incorporating this feature, I enhanced the model's ability to account for the combined effect of these variables on delay durations.

Data Preprocessing and Outlier Handling

Looking at the initial high RMSE of 45.41, I noticed that outliers in the target variable (ArrDelayMinutes) were negatively affecting the model's performance.To address this, I capped extreme values in the target variable at the 95th percentile. This step reduced the impact of rare, extreme delays, allowing the model to focus on the majority of the data and generalize better.

To prepare the feature set for modeling, I applied feature scaling using StandardScaler to standardize the values across all features. Since linear regression is sensitive to the magnitude of feature values, scaling ensured that no single feature dominated the model due to its scale.

After training, I evaluated the model using RMSE and R^2 metrics. Initially, the model's RMSE was 45.41, with an R^2 score of 0.07, indicating significant room for improvement. After implementing the interaction feature, scaling, and outlier capping, the model showed substantial improvement. The RMSE dropped to 15.60, and the R^2 score increased to 0.21, which indicated that the applied techniques significantly enhanced the model's predictive power.

Feature Importance Analysis

To better understand the model's performance and provide actionable insights, I analyzed feature importance by examining the coefficients of the linear regression model. The interaction feature, Distance_AvgDelays, was one of the most influential predictors, validating its inclusion in the model.

Residual Analysis

Finally, I examined the residuals (the difference between actual and predicted delay durations) to evaluate the model's behavior further. The residual distribution after improvements showed reduced variability, with most errors clustering around zero. This confirmed that the model was making more accurate predictions and better generalizing to unseen data.

I applied machine learning principles such as feature engineering, outlier handling, feature scaling, and model evaluation to develop a better performing regression model. These steps improved the model's ability to predict flight delays and provided stakeholders with actionable insights into delay patterns.

[Naima Amraan]: Evaluation of Logistic Regression Model and Tuning

Confusion Matrix Analysis
To understand the initial performance of the logistic regression model, I analyzed the confusion matrix. This revealed the following key values:

- **True Negatives (TN)**: 781,762 cases correctly classified as the negative class.
- **False Positives (FP)**: 241,800 cases incorrectly classified as positive.
- **False Negatives (FN)**: 58,501 cases incorrectly classified as negative.
- **True Positives (TP)**: 155,111 cases correctly classified as positive.

These results indicated that while the model was effective in identifying negative cases, it struggled with false positives and false negatives. This imbalance demonstrated the need for further optimization, particularly in balancing precision and recall to reduce classification errors.

Threshold Optimization
To improve the balance between precision and recall, I optimized the classification threshold. Initially, the model used the default threshold of **0.5**, which produced the following metrics:

- **Accuracy**: 75.73%
- **Precision**: 39.08%
- **Recall**: 72.61%
- **F1 Score**: 0.5081

By adjusting the threshold to **0.6217**, I maximized the F1 score and achieved significant improvements:

- **Accuracy**: Improved from **75.73%** to **84.05%**.
- **Precision**: Improved from **39.08%** to **53.63%**, indicating fewer false positives relative to true positives.
- **Recall**: Decreased slightly from **72.61%** to **56.34%**, reflecting the trade-off of prioritizing precision over recall.
- **F1 Score**: Improved from **0.5081** to **0.5495**, showing a better balance between precision and recall.

The improvement in accuracy to **84.05%** was a direct result of threshold tuning, as this adjustment directly impacts classification metrics by altering the balance between false positives and false negatives.

Hyperparameter Tuning
I conducted hyperparameter tuning using **RandomizedSearchCV**. This tuning aimed to optimize the **ROC-AUC score**, to evaluate the model's ability to distinguish between classes independently of thresholds.
I optimized the following hyperparameters:

- **Solver**: 'liblinear', a solver suitable for smaller datasets and regularization methods.
- **Penalty**: 'l2', which applies ridge regularization to prevent overfitting.
- **Class Weight**: 'balanced', which adjusts weights inversely proportional to class frequencies to address class imbalance.
- **Regularization Strength (C)**: A grid of [0.1, 1] was tested, representing varying levels of regularization.

The best parameters identified were (**Solver**: 'liblinear', **Penalty**: 'l2', **Class Weight**: 'balanced', and **C**: 1

I used 2-fold cross-validation and evaluated 3 parameter combinations. The best cross-validated ROC-AUC score achieved was 0.6713, showing a notable decrease compared to the original ROC-AUC of 0.8164. On the test set, the ROC-AUC score was similar, at 0.6728.

Metrics Evaluation
To comprehensively evaluate the model's performance, I used the following metrics:

1. **Accuracy**: To assess overall correctness of predictions.
2. **Precision**: To minimize false positives.
3. **Recall**: To minimize false negatives.
4. **F1 Score**: To balance precision and recall.
5. **ROC-AUC**: To measure the model's discriminative ability across thresholds.

Threshold Tuning Results
By comparing the baseline threshold of **0.5** to the optimized threshold of **0.6217**, I observed the following improvements:

- **Accuracy**: Increased from **75.73%** to **84.05%**.
- **Precision**: Increased from **39.08%** to **53.63%**, reducing false positives.
- **F1 Score**: Increased from **0.5081** to **0.5495**, indicating better balance between precision and recall.

However, **recall** decreased slightly, dropping from **72.61%** to **56.34%**, reflecting the trade-off made to prioritize precision.

Observations and Recommendations

- **Improvements Achieved**:
  - Threshold tuning significantly improved accuracy from **75.73%** to **84.05%**, precision from **39.08%** to **53.63%**, and F1 score from **0.5081** to **0.5495**.
- **Limitations**:
  - Recall decreased slightly during threshold tuning, reflecting the trade-off made to prioritize precision.
  - The **ROC-AUC score** decreased substantially during hyperparameter tuning from the initial score of **0.8164**, possibly due to removing features directly related to delays.
- **Trade-Offs**:
  - The reduction in ROC-AUC and accuracy highlights the impact of excluding delay-related features, which could have been strong predictors of the target variable.
  - These results suggest that logistic regression may struggle to achieve high performance with the current feature set.

[Diana Chen]: Evaluation of Decision Tree Model and Tuning

To evaluate our Decision Tree model, I used key binary classification metrics: accuracy, precision, recall, and F1-score, but focused on those associated with the psychological impact of prediction outcomes on stakeholders to align model objectives with user needs. [2]

Error Type	Consideration & Potential Repercussions	Metric
True Positives	(Accurate Delay Notifications) Passengers are informed of a delay that occurs as predicted. This builds or maintains trust in the airline's communication.	Recall (Sensitivity)
False Positives	(False Alarms) Passengers are notified of a delay that does not materialize, potentially causing frustration and reducing trust in communication accuracy.	Precision
False Negatives	**(Missed Delay Notifications) Passengers experience unexpected delays without prior notification, leading to significant dissatisfaction and perceived mismanagement.**	**Recall**
True Negatives	(On-Time Flights Without Notification) Flights operate as scheduled without delay notifications, maintaining standard satisfaction levels.	Accuracy

Evaluation Methods & Tuning Techniques

Confusion Matrix, Classification Report, Cross-Validation
Across all the model refinement, I utilized Classification Reporting, Confusion Matrices, and ROC/Precision-Recall curves to visualize model performance. In addition to reviewing the metrics, I used cross-validation across 5-folds to check its performance across multiple splits. The initial baseline model resulted in these metrics:

- **Accuracy:** 0.6524522823790349
- **Precision:** 0.26675191705739765
- **Recall:** 0.5809651964760436
- **F1 Score:** 0.36562569618927204
- **ROC-AUC Score:** 0.6665443466873683

Undersampling for an Imbalanced Dataset
To address class imbalance and improve performance (precision and recall) for the minority class, I used resampling and class weight adjustment techniques. I initially considered oversampling the minority class, but because of the large size and dimensionality of the dataset, I found undersampling the majority class to be the better and more feasible approach to addressing the imbalance.

Class Weight Adjustment
In continuing to attempt to refine for the imbalanced data, I adjusted `class_weight` and experimented with custom weights to add a stronger penalty on misclassifications of class 1

delays. To start the process, I considered the imbalance ratio to inform the weight adjustments and tweaked from there.

Hyperparameter Optimizing
Due to the significant skew towards the majority class, I was concerned about the potential for overfitting to the majority class during model training. To address this, I performed hyperparameter tuning on the Decision Tree classifier, focusing on key parameters such as `max_depth`, `min_samples_split`, and `min_samples_leaf`. These parameters were chosen to control overfitting and underfitting by limiting tree complexity and ensuring splits occurred only when meaningful patterns could be identified.

I employed grid and random search techniques to identify optimal hyperparameter combinations. However, these methods proved computationally expensive due to the dataset's size and dimensionality. Despite the exhaustive search, the tuned parameters did not yield notable improvements in precision or recall, indicating that generalization efforts were less impactful than the earlier class rebalancing. This suggests that class imbalance played a more critical role in model performance than hyperparameter optimization for this dataset.

Feature Engineering
Based on the shifts in top 20 features between models, I explored some additional feature engineering, such as one-hot encoding month and day, but the improvements were minimal.

Observations & Limitations

Ultimately, adjustments to the decision tree model were not able to capture enough of the complexities, as seen in the low recall and precision scores and consistently low F-1 scores. There were tradeoffs between recall and precision. Given the critical stakeholder impact of false negatives, recall was prioritized as the key metric to minimize missed delay notifications.

- **Accuracy:** 0.6443984435495734
- **Precision:** 0.2722075938172097
- **Recall:** 0.634959185682872
- **F1 Score:** 0.38105608805939284
- **ROC-AUC Score:** 0.6960616836652348

While the best overall are seen above, there were model versions with clear tradeoffs between precision and recall. In particular, adding penalties to misclassifications of positives increased recall but at the expense of decreased precision and accuracy.

[Maya Schoucair]: Evaluation of Random Forest Model and Tuning

My contribution to this final project was evaluating and tuning the random forest model and its hyperparameters. There were three versions of the model I evaluated - the baseline model, the baseline model after feature selection, and the hyperparameter tuned model.

Baseline Model Evaluation:

To start, I evaluated the performance of the baseline random forest model to get a better understanding of where improvements could be made. The baseline model had very high evaluation metrics. The **classification report** showed an accuracy of 95%, a precision of 86%, a recall of 88%, and a F-1 score of 87%. The **confusion matrix** showed the following:

- **True Negatives (TN)**: 991,644 cases correctly classified as negative.
- **False Positives (FP)**: 31,918 cases incorrectly classified as positive
- **False Negatives (FN)**: 24,668 cases incorrectly classified as negative
- **True Positives (TP)**: 188,944 cases correctly classified as positive

Similarly, the **AUC** was excellent, with a value of 0.99. However, this extremely high value led me to believe that there may have been some data leakage that occurred, leading to the model's performance being artificially inflated.

Feature Selection & Re-Evaluation:

After reviewing the features used in the baseline model as well as their importance values, I dropped features that would not be possible to know in real-time (at prediction time), such as 'ArrTime,' 'DepTime,' and 'ActualElapsedTime' to eliminate data leakage. Additionally, I dropped features that were highly correlated with one another, such as 'Quarter' and 'CRSArrTime' to avoid collinearity. As suspected, after removing those features, all of the model's evaluation metrics declined. The **classification report** showed an accuracy of 75%, a precision of 36%, a recall of 54%, and a F-1 score of 43%. The AUC dropped to 0.74 and the **confusion matrix** showed the following:

- **True Negatives (TN)**: 817,513 cases correctly classified as negative.
- **False Positives (FP)**: 206,049 cases incorrectly classified as positive
- **False Negatives (FN)**: 97,285 cases incorrectly classified as negative
- **True Positives (TP)**: 116,327 cases correctly classified as positive

Hyperparameter Tuning & Cross-Validation:

To attempt to improve upon the revised model, I conducted **RandomizedSearchCV** to tune its hyperparameters to optimize for ROC-AUC and evaluate its performance on unseen data via two-fold cross validation. The hyperparameters I tested were: 'n_estimators', 'max_depth', 'min_samples_split', 'min_samples_leaf', and 'amx_features'. The best hyperparameters were 200, 5, 2, 'sqrt', and None respectively.

I also conducted **RandomizedSearchCV** to tune its hyperparameters to optimize for F-1 scores with the same hyperparameters being tested. The best hyperparameters in this case were 200, 2, 1, ‘log2’, and 20 respectively.

Threshold Optimization & Final Model Evaluation:

Using the hyperparameters from the first round of **RandomizedSearchCV** (that optimized for ROC-AUC), I re-trained a random forest classifier with the same features as the second model (baseline model after feature selection). I opted against re-training a model with the second set of hyperparameters (that optimized for F-1 score) due to the extremely high computational cost of modeling on our large dataset. Instead, I found the **optimal decision threshold** that would give me the best F-1 score from my tuned model. The best threshold was 0.44, which gave me a F-1 score of 45%. At this threshold, the accuracy was 69%, precision was 33%, and recall was 79%.

[Elsie David] : Regression Model Hyperparameter Tuning and Evaluation

My contribution to this project focused on the regression model's hyperparameter tuning and evaluation which involved refining the models, optimizing its parameters and analyzing feature importance to improve predictions.

1. **Model Selection and Setup**
    - I implemented Ridge and Lasso regression models for their ability to handle multicollinearity and perform feature selection.
    - Linear models were chosen for their interpretability and for regularization purposes which was important for this high dimensional dataset with potentially redundant features.
2. **Hyperparameter Optimization**
    - I first used a grid search function to optimize the alpha parameter (which controls the regularization strength) but found it to be computationally expensive because of the exhaustive parameter testing so I used RandomizedSearchCV instead, which sampled from a uniform distribution of alpha values.
    - 5 fold cross-validation was implemented in this process to enhance performance across different subsets of the data.
3. **Data Sampling for Efficiency**
    - Because of the computational cost of training on the 80% training set (it kept running for too long and crashing eventually at first), I sampled the training set and trained the models on a 50% sample, the test set was retained for unbiased evaluation. This helped ensure faster computations with minimal impact on model performance.
4. **Model Evaluation**
    - Evaluated the models using the following metrics:
        - **Root Mean Squared Error (RMSE)**: To penalize large errors in predictions.
        - **Mean Absolute Error (MAE)**: To measure the average magnitude of errors.
        - **R^2 Score**: To assess the proportion of variance explained by the model.
5. **Feature Selection**
    - I identified and removed highly correlated features to address multicollinearity and improve the model's stability. I also excluded some features like DepTime and ArrTime, which are not available before a flight delay happens, to prevent data leakage and ensure the model is aligned with real world scenarios. I iteratively refined feature sets and checked their impact on model performance before arriving at the final feature set.
6. **Polynomial Features**
    - To capture non-linear relationships, I expanded the top 20 most predictive features (based on the Lasso regression feature importance) into second degree polynomial features. This allowed the model to account for interactions between

variables and higher order effects. The expanded feature set was standardized and used with Ridge regression which resulted in a slight performance in improvement.

Challenges and Solutions

- The large dataset posed computational challenges, especially during polynomial feature expansion. In order to address this, I:
    - Subsampled the training data to 50%.
    - Focused on the most relevant features to reduce dimensionality.
- Polynomial feature interactions improved model complexity but required careful scaling and tuning to avoid overfitting and to manage compute resources.

RESULTS & CONCLUSIONS

Model 1 Results

- The three models for predicting the binary outcome of whether a flight would be delayed or not showed different levels of performance. Due to the class imbalance in our dataset, the models experienced large tradeoffs in their evaluation metrics. For example, a model with high recall often had low precision. Below, we summarize the metrics for our three models with the highest recall score respectively. We opted to prioritize recall because we perceive false negatives to be more costly to both passengers and airlines than false positives.
    - Logistic regression model:
        - Accuracy: 0.6253704006065436
        - Precision: 0.2586328808726194
        - **Recall: 0.6267063648109656**
        - F1 Score: 0.3661576585077664
    - Decision tree model:
        - Accuracy: 0.6443984435495734
        - Precision: 0.2722075938172097
        - **Recall: 0.634959185682872**
        - F1 Score: 0.38105608805939284
    - Random forest model:
        - Accuracy: 0.6958228996083009
        - Precision: 0.3375071405459172
        - **Recall: 0.7910463831619946**
        - F1 Score: 0.47314335955109665

Model 2 Results

- The optimized regression models showed different levels of performance in predicting the target variable (in this case it is arrival delays in minutes). The Polynomial Ridge model performed best with an **RMSE of 16.87**, **MAE of 11.10** and an **R^2 score of 0.07**, indicating slight improvements in predictive accuracy by capturing non-linear relationships. This was followed by **Ridge Regression** with an **RMSE of 17.03**, **MAE of 11.27** and an **R^2 score of 0.057**. Finally, the **Lasso Regression** model had an **RMSE of 17.06**, **MAE of 11.35** and an **R^2 score of 0.054**. These results demonstrate that the models were only able to explain 5–7% of the variance in arrival delays which is quite low, suggesting that non-linear modeling approaches may be necessary to improve predictive performance.

Real World Implications

With an average of over 100,000 commercial flights operating per day, the implications of our project are far reaching [2]. Having access to reliable predictions regarding flight status would have significant benefits for an increasingly interconnected world, such as allowing airlines to optimize their resource allocation and passengers to optimize their schedules.

Despite the benefits that would come from being able to predict flight delays, in reality, doing so can be very challenging. The travel and aviation industries are extremely unpredictable, affected by factors such as weather, maintenance or personnel issues, and global events. None of this information was available in our dataset and much of it is likely to be unavailable in real-world predictions as well. Delays may also be influenced by complex relationships that could not be captured by the modelling approach. For example, no one could have predicted the 2024 Crowdstrike outage that led to over 29,000 flight delays worldwide [4]. Thus, while predictive modeling can provide stakeholders with some insights, the predictions are made in a highly dynamic and volatile problem space.

Additionally, the benefits of such predictions must be weighed against the costs, particularly when the predictions are incorrect. The costs of faulty predictions can fall on passengers, airline employees, and the airlines themselves. For example, a false positive (i.e. a prediction that a flight will be delayed when it is not) may lead passengers to miss their flight or airlines to prematurely rebook passengers who have connecting flights. A false negative on the other hand (i.e. failing to accurately predict a delay), could lead passengers to miss connecting flights and worsen disruptions. It is clear that in a high-stakes industry with a low tolerance for error, predictions may cause more harm than good.

To improve predictive power, real-time data is necessary. With real-time information regarding factors such as weather, technical issues, and air traffic (amongst others), a model would be better able to make accurate predictions regarding flight delays. Additionally, incorporating live data would allow for an adaptive machine learning model to be used, resulting in more responsive and accurate real-time predictions regarding one’s flight status.

REFERENCES

[1] Li, W., Correia, A. R., & Nijkamp, P. (2021). Customer Trust and Satisfaction with Airline Services: The Role of Delay Management Communication. Journal of Air Transport Management, 92, 102038.

[2] C.-J. Lin, “A Practical Guide to Support Vector Classification,” [Online]. Available: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[3] “Airline Frequency and Capacity Statistics | Aviation Data | OAG,” www.oag.com. https://www.oag.com/airline-frequency-and-capacity-statistics]

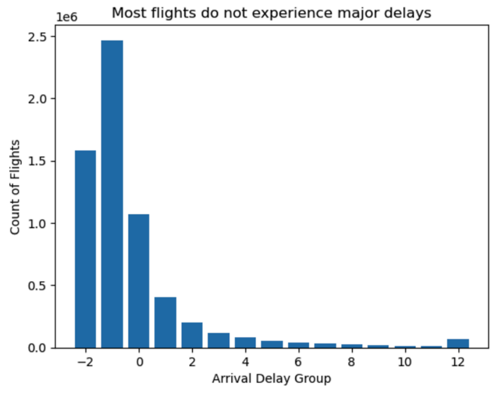
[4] F. International, “CrowdStrike Bug Leads to Global Outages and Flight Delays,” Fiu.edu, Jul. 22, 2024. https://gordoninstitute.fiu.edu/news-events/the-policy-spotlight/2024/crowdstrike-bug-leads-to-global-outages-and-flight-delays.html

APPENDIX

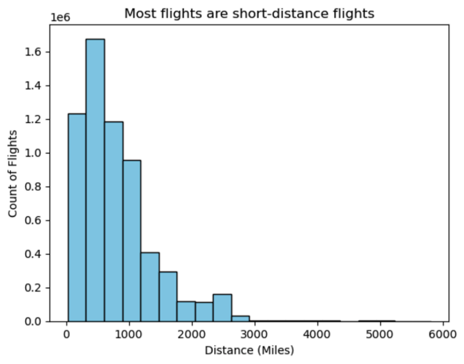
Class imbalance of binary target variable ‘ArrDel15’:

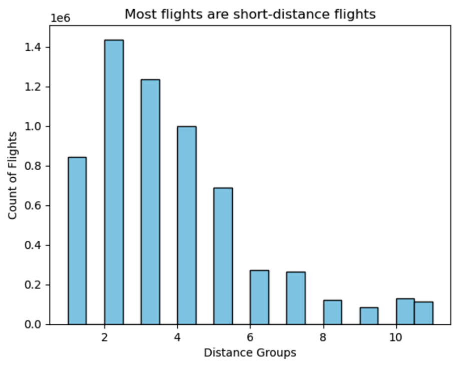
```
0.0    5117811
1.0    1068059
Name: ArrDel15, dtype: int64
```

Right skewed distribution of ‘ArrivalDelayGroups’. Most flights do not have major delays:



Right skewed distribution of flight ‘Distance’:

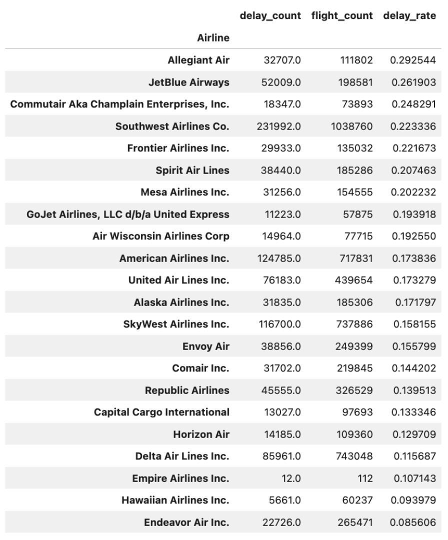




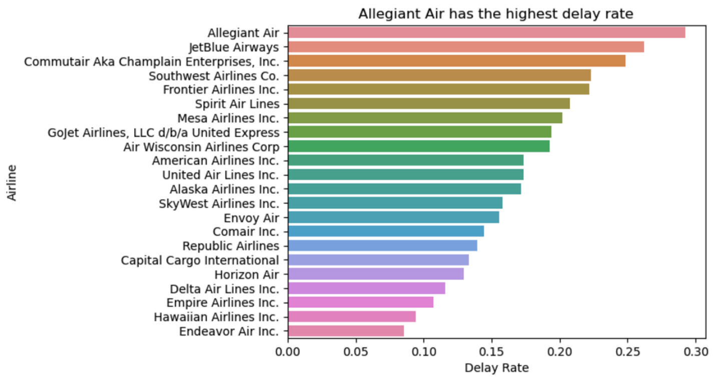
Diverse set of unique values for non-numerical variables:

variable_name	unique_value_count
Operating_Airline	22
Airline	22
OriginStateName	53
DestStateName	53
OriginCityName	374
DestCityName	374
Origin	380
Dest	380
Tail_Number	5758

Allegiant Air has the highest delay rate, followed by JetBlue Airways:

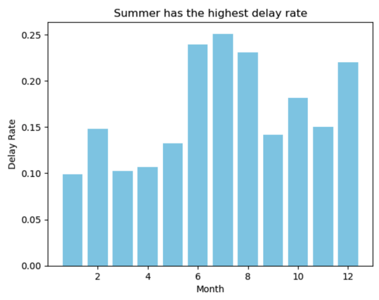


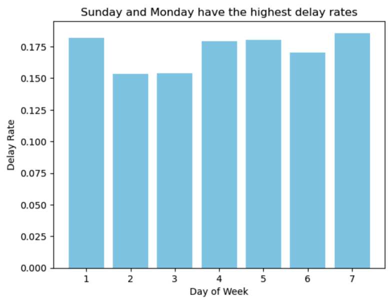
Airline	delay_count	flight_count	delay_rate
Allegiant Air	32707.0	111802	0.292544
JetBlue Airways	52009.0	198581	0.261903
Commutair Aka Champlain Enterprises, Inc.	18347.0	73893	0.248291
Southwest Airlines Co.	231992.0	1038760	0.223336
Frontier Airlines Inc.	29933.0	135032	0.221673
Spirit Air Lines	38440.0	185286	0.207463
Mesa Airlines Inc.	31256.0	154555	0.202232
GoJet Airlines, LLC d/b/a United Express	11223.0	57875	0.193918
Air Wisconsin Airlines Corp	14964.0	77715	0.192550
American Airlines Inc.	124785.0	717831	0.173836
United Air Lines Inc.	76183.0	439654	0.173279
Alaska Airlines Inc.	31835.0	185306	0.171797
SkyWest Airlines Inc.	116700.0	737886	0.158155
Envoy Air	38856.0	249399	0.155799
Comair Inc.	31702.0	219845	0.144202
Republic Airlines	45555.0	326529	0.139513
Capital Cargo International	13027.0	97693	0.133346
Horizon Air	14185.0	109360	0.129709
Delta Air Lines Inc.	85961.0	743048	0.115687
Empire Airlines Inc.	12.0	112	0.107143
Hawaiian Airlines Inc.	5661.0	60237	0.093979
Endeavor Air Inc.	22726.0	265471	0.085606

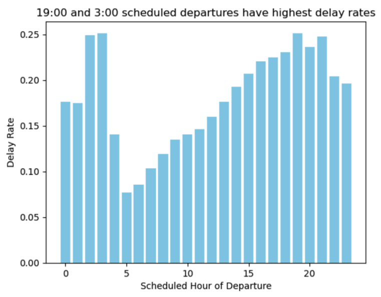


Delay Rate by Flight Time:

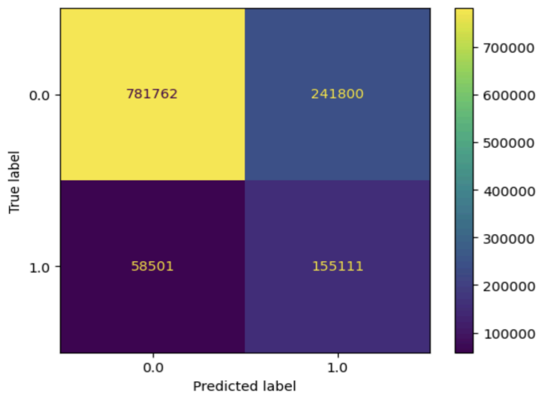
a. Summer has the highest delay rate.
b. Sunday and Monday have the highest delay rates, but other days of week show similar delay rates.
c. 19:00 and 3:00 scheduled departure flights have the highest delay rates.



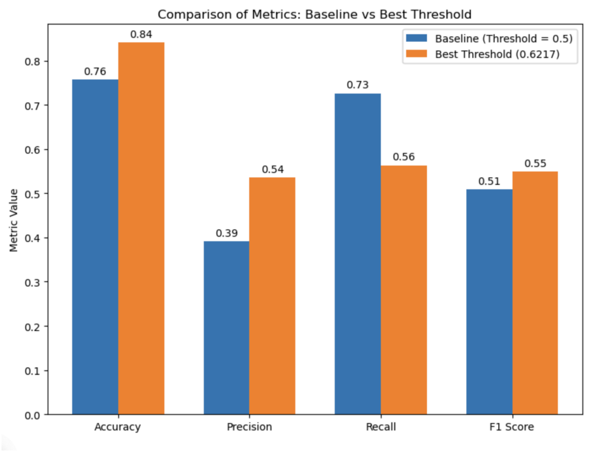




7. Confusion matrix: To understand the initial performance of the logistic regression model



8. Threshold comparisons for tuning logistic regression model


Comparison of Metrics: Baseline vs Best Threshold
Baseline (Threshold = 0.5)
Best Threshold (0.6217)
Metric Value
Accuracy
Precision
Recall
F1 Score
0.76
0.84
0.39
0.54
0.73
0.56
0.51
0.55
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8