Milestone 2

Airline Delay Prediction

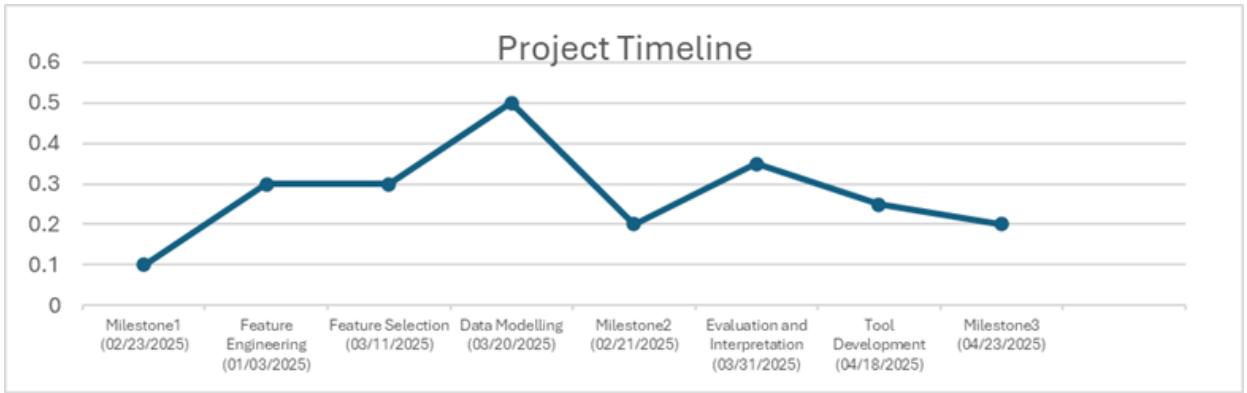
Objective:

The goal of this project is to develop an interactive dashboard that leverages machine learning models to accurately predict flight delays, using various datasets such as the U.S. International Air Traffic data, US Domestic Flights, Airline codes data, Storm Events data, and Consumer Airfare Reports. By analyzing flight operations, delays, and external factors like weather events and airfare trends, the objective is to identify key contributors to flight delays and provide insights that can improve airline operational efficiency. The predictive model will be built on historical data to forecast future flight delays, enabling airlines and passengers to make more informed decisions.

Tech Stack:

- **Python**: Primary programming language used for data processing and modeling.
- **Pandas, NumPy, SciPy, sklearn**: Libraries for data manipulation, preprocessing, and statistical analysis.
- **SQLite**: Database used to store cleaned and structured data.
- **Matplotlib, Seaborn, Plotly**: Tools for data visualization and exploratory data analysis (EDA).
- **Scikit-Learn, PyTorch**: Frameworks for building and training machine learning models.
- **Flask**: API framework used to serve predictions from the model.
- **Streamlit**: Platform for building an interactive web-based dashboard to display model insights.

Project Timeline:



Dataset Description:

Name	Source	Attributes	Description	ntributor
International Air Traffic data(1990-2020)		'Year', 'Month', 'usg_apt_id', 'usg_apt', 'fg_apt_id', 'fg_apt', 'fg_wac', 'carrier', 'carriergroup', 'type', 'Scheduled', 'Charter', 'Total' 'Year', 'Month', 'usg_apt_id', 'usg_apt', 'fg_apt_id', 'fg_apt', 'fg_wac', 'carrier', 'carriergroup', 'type', 'Scheduled', 'Charter', 'Total'	ures: Data on all flights n US gateways and non-US ys, irrespective of origin stination. bservation provides ation on a specific airline air of airports, one in the US e other outside. Three main s record the number of Scheduled, Charter, and gers: Data on the total r of passengers for each and year between a pair of s, as serviced by a particular	
ht Delay and Cancellation Dataset (2019-2023)	Kaggle	, 'AIRLINE', 'AIRLINE_DOT', CODE', 'DOT_CODE', 'FL_NUMBER', 'ORIGIN', ITY', 'DEST', 'DEST_CITY', 'CRS_DEP_TIME', 'DEP_TIME', AY', 'TAXI_OUT', 'WHEELS_OFF', 'WHEELS_ON', 'TAXI_IN', _TIME', 'ARR_TIME', 'ARR_DELAY', 'CANCELLED', ATION_CODE', 'DIVERTED', PSED_TIME', 'ELAPSED_TIME', 'AIR_TIME', E', 'DELAY_DUE_CARRIER', 'DELAY_DUE_WEATHER', UE_NAS', 'DELAY_DUE_SECURITY', 'DELAY_DUE_LATE_AIRCRAFT'	s a collection of flight metrics for commercial cludes information on d actual flight operations, dates, carrier codes, flight airport details (origin and The dataset captures crucial ata including scheduled and ure/arrival times, delays d arrival), taxi times, and time. Additionally, it iled breakdowns of delay as carrier-related, ed, and security delays—as tors for cancellations and	

Storm Events data	**ional Centers for mental Information**	EARMONTH', 'BEGIN_DAY', ME', 'END_YEARMONTH', 'END_DAY', 'END_TIME', ID', 'EVENT_ID', 'STATE', 'STATE_FIPS', 'YEAR', 'MONTH_NAME', YPE', 'CZ_TYPE', 'CZ_FIPS', ', 'WFO', 'BEGIN_DATE_TIME', ZONE', 'END_DATE_TIME', 'INJURIES_DIRECT', _INDIRECT', 'DEATHS_DIRECT', 'DEATHS_INDIRECT', _PROPERTY', 'DAMAGE_CROPS', 'MAGNITUDE', 'MAGNITUDE_TYPE', AUSE', 'CATEGORY', 'TOR_F_SCALE', 'TOR_LENGTH', 'TOR_WIDTH', ER_WFO', 'TOR_OTHER_CZ_STATE', 'TOR_OTHER_CZ_FIPS', ER_CZ_NAME', 'BEGIN_RANGE', 'BEGIN_AZIMUTH', CATION', 'END_RANGE', MUTH', 'END_LOCATION', 'BEGIN_LAT', N', 'END_LAT', 'END_LON', 'EPISODE_NARRATIVE', ARRATIVE', 'DATA_SOURCE'	Events dataset etailed information weather occurrences ted States. It includes ng, location (state, ographic s), event type, and trics such as injuries, and damage estimates. et is used to analyze atterns and assess the severe storms	
Airline Fleets		Airline', 'Airline', 'Aircraft Type', 'Current', 'Future', 'Historic', 'Total', 'Orders', 'Unit tal Cost (Current)', 'Average Age'	et includes details on ines, individual nds, aircraft types, erating fleets, future it and total aircraft the average age of the	

nsumer Airfare Report	[Data.Gov]	r', 'quarter', 'mkt_fare', etid_1', 'citymarketid_2', 'city1', 'city2', eid', 'car', 'carpax', 'carpaxshare', 'caravgfare', min', 'fareinc_minpaxsh', 'fareinc_max', _maxpaxsh', 'fare_inc_x3paxsh', 'Geocoded_City1', d_City2', 'tbl5pk'	et includes metrics arket fare, year, ty market identifiers, ne identifiers, metrics, fare s, and geocoded tails—supporting f domestic short-haul	
Airline IATA code	[Wikipedia]	CAO', 'Airline', 'Call sign', 'Region', 'Comments'	napping of airline names	

I. Feature Engineering:

Time-Based Features:

In this feature engineering task, we focused on creating time-based features that capture the impact of flight departure and arrival times on potential flight delays. The steps undertaken were as follows:

1. Conversion of Scheduled Departure and Arrival Times

Feature: Dep_Hour, Arr_Hour
Justification:
The original flight times were provided in HHMM format, which, while useful for displaying exact times, does not easily allow for analysis of flight patterns based on time-of-day. By extracting the hour of the day from the scheduled departure and arrival times, we make it easier to analyze trends and identify correlations between flight performance and different hours of the day. The features Dep_Hour (Scheduled Departure Hour) and Arr_Hour (Scheduled Arrival Hour) allow us to focus on the time dimension and examine how specific hours might correlate with delays or cancellations.

2. Categorization of Time into Time-of-Day Bins

Feature: DepTimeOfDay, ArrTimeOfDay
Justification:
Flight delays and cancellations may vary depending on the time of day, as different periods could be impacted by different factors such as airport congestion, weather patterns, or operational procedures. By categorizing flight departure and arrival times into broader time-of-day bins (morning, afternoon, evening,

and night), we can capture these time-related patterns and identify whether certain times of day are more prone to delays. This categorization is especially useful for capturing the effects of peak and off-peak hours, which could significantly influence flight performance.

3. Day of the Week Feature

Feature: DayOfWeek
Justification:
Flight performance might be affected by the day of the week, as certain days (e.g., weekdays versus weekends) may have varying levels of air traffic, staffing, and operational conditions. By extracting the day of the week from the FlightDate column, we gain insight into whether specific days have a higher frequency of delays or cancellations. This feature is essential for identifying weekly patterns and understanding whether certain days are more prone to flight disruptions due to factors like increased passenger volume or staff availability.

Damage Based Features:

1. Combination of Direct and Indirect Injuries

   Feature: Total_Injuries

   Justification:

   The dataset contained separate columns for direct injuries (INJURIES_DIRECT) and indirect injuries (INJURIES_INDIRECT). To obtain a more holistic view of the total number of injuries resulting from an event, these two columns were summed to create the Total_Injuries feature. This aggregated feature allows for easier analysis of the total injury impact of an event, without distinguishing between the source of the injury.

2. Combination of Direct and Indirect Deaths

   Feature: Total_Deaths

   Justification:

   Similar to the injuries, the dataset contained separate columns for direct deaths (DEATHS_DIRECT) and indirect deaths (DEATHS_INDIRECT). These two variables were combined into a single feature, Total_Deaths, to provide a clearer picture of the overall mortality impact of the event. Aggregating these values allows for a more straightforward assessment of the total number of deaths, without having to distinguish between direct and indirect causes.

3. Combination of Property and Crop Damage

Feature: Total_Damage

Justification:

The DAMAGE_PROPERTY and DAMAGE_CROPS columns represent different types of economic damage (property and crops) resulting from an event. To get a more comprehensive view of the total financial impact of the event, these two damage values were combined into a single feature, Total_Damage. This aggregated feature simplifies the analysis by providing a single measure of total damage, regardless of whether it pertains to property or crops.

Flight Traffic Based features:

1. Binning Flight Distance into Categories

Feature: Distance_Bins_miles, IsShortFlight, IsMediumFlight, IsLongFlight

Justification:

The Distance_miles column, which represents the distance of the flight, was converted into categorical bins to classify the flight journey into "short", "medium", or "long" distances. This transformation helps to simplify the analysis by grouping flights based on their journey length, which can impact delay patterns.

- Short flights are those under 250 miles.
- Medium flights range from 250 to 750 miles.
- Long flights are those over 750 miles.

These bins can help identify patterns in flight performance that may vary depending on journey length. To facilitate machine learning models, binary indicator features (IsShortFlight, IsMediumFlight, IsLongFlight) were created to flag whether a flight belongs to a particular bin

2. Delay and Distance Interaction

Feature: Delay_Distance_Interaction

Justification:

A new feature, Delay_Distance_Interaction, was created by multiplying the arrival delay (Arr_Delay_min) by the distance (Distance_miles). This interaction term allows us to capture the combined effect of delay and distance on flight performance. Longer flights may experience delays due to a variety of factors, such as air traffic, weather, and flight duration. By creating this feature, we aim to assess whether longer flights tend to experience higher delays or if delays are more significant for shorter flights.

3. Route Encoding

   Feature: Route

   Justification:

   The combination of the origin and destination states was used to create a new feature, Route, which represents the flight's route as a concatenation of the origin and destination states (e.g., 'NY-CA'). This feature allows us to examine flight performance at the route level, which may reveal patterns such as consistently delayed routes due to specific factors like weather or air traffic. Encoding the route as a single categorical feature also allows it to be used in predictive models or aggregation operations.

4. Average Delays per Airline and Route

   Features: AvgDepDelayByAirline_min, AvgArrDelayByAirline_min, AvgDepDelayByRoute_min, AvgArrDelayByRoute_min

   Justification:

   To capture airline-specific and route-specific trends in delays, average departure and arrival delays were calculated per airline and per route. These averages provide insight into how an airline or route performs on average with respect to delays. For example, if an airline or route consistently experiences higher delays, this information can be used to adjust expectations or make improvements to operations.

   These features were generated by grouping the dataset by Airline_Code and Route, calculating the mean delay for each group, and then merging these statistics back into the main dataset. The resulting features were renamed appropriately to reflect whether the delays were departure or arrival delays and whether they were grouped by airline or route

5. Route Traffic Feature

Feature: RouteTraffic

Justification:

The RouteTraffic feature was created to capture the frequency of flights on a specific route on a given day. By counting the number of flights per route and date, this feature reflects how busy a particular route is on a given day. High traffic routes might be more prone to delays due to congestion, while routes with lower traffic could see fewer delays. This feature can provide valuable context for understanding delays in relation to traffic patterns.

Weather Based Features:

1. Severe Weather Flag

Feature: SevereWeatherFlag

Justification:

The SevereWeatherFlag feature was created to flag severe weather conditions based on the MAGNITUDE column. If any weather event recorded for a destination state on a specific flight date has a magnitude greater than 5, it is considered severe, and the flag is set to 1. If the magnitude is 5 or below, the flag is set to 0. This feature provides a binary indicator that signals whether severe weather might have impacted flight performance on that particular day.

Severe weather can be a critical factor influencing delays, cancellations, or other flight disruptions. By including this flag, the dataset is enriched with a useful feature that directly accounts for the potential impact of weather on flights.

Encoding Categorical Variables:

Justification: Categorical columns were encoded based on the number of unique values they contain. The decision between using One-Hot Encoding (OHE) or Label Encoding was determined by a threshold value. If the categorical column has a number of unique values less than or equal to the threshold (3), One-Hot Encoding was applied. If the number of unique values exceeds the threshold, Label Encoding was used.

One-Hot Encoding (OHE): This technique is applied to categorical variables that have a small number of distinct categories. It creates a binary column for each unique value in the categorical variable. For instance, if a column like DepTimeOfDay has 3 categories ('morning', 'afternoon', 'evening'), OHE will create three columns, each representing one category as a binary indicator (0 or 1). This is particularly useful when the variable is nominal, meaning the categories do not have an ordinal relationship.

Label Encoding: This technique is applied when the categorical column has a larger number of unique values (more than 3 in this case). It converts each category into a numerical label, allowing the machine learning model to work with numeric values. For instance, an Airline_Code column containing airline codes like 'AA', 'DL', 'UA' will be encoded into numerical values like 0, 1, and 2, respectively. This is useful when the variable represents an ordinal or hierarchical relationship, or when there are many unique categories.

Categorical Columns Encoded:

The following columns were encoded based on the specified criteria:

· Airline_Code: The airline code was Label Encoded as it has more than 3 unique values.

· OriginState and DestState: These state variables were One-Hot Encoded since the number of unique states is small.

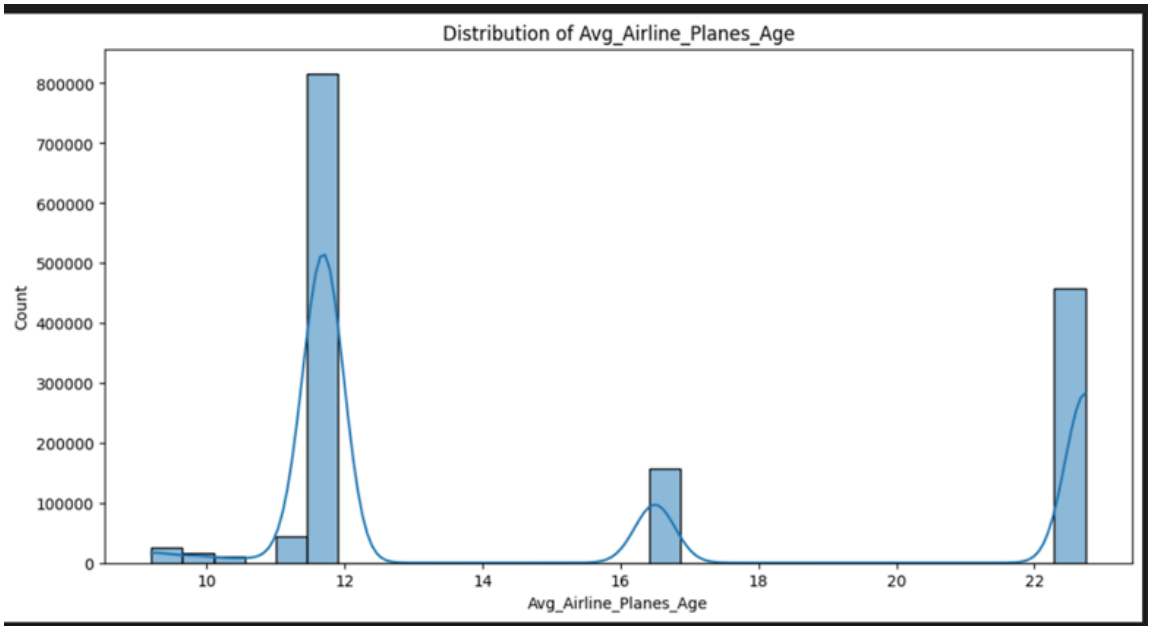
· Origin_City and Dest_City: Cities were One-Hot Encoded due to a smaller number of unique cities.

· DepTimeOfDay and ArrTimeOfDay: These time-of-day features were One-Hot Encoded since they likely have limited values (e.g., morning, afternoon, evening, night).

· Route: This feature was One-Hot Encoded as it is more likely to have fewer distinct values.

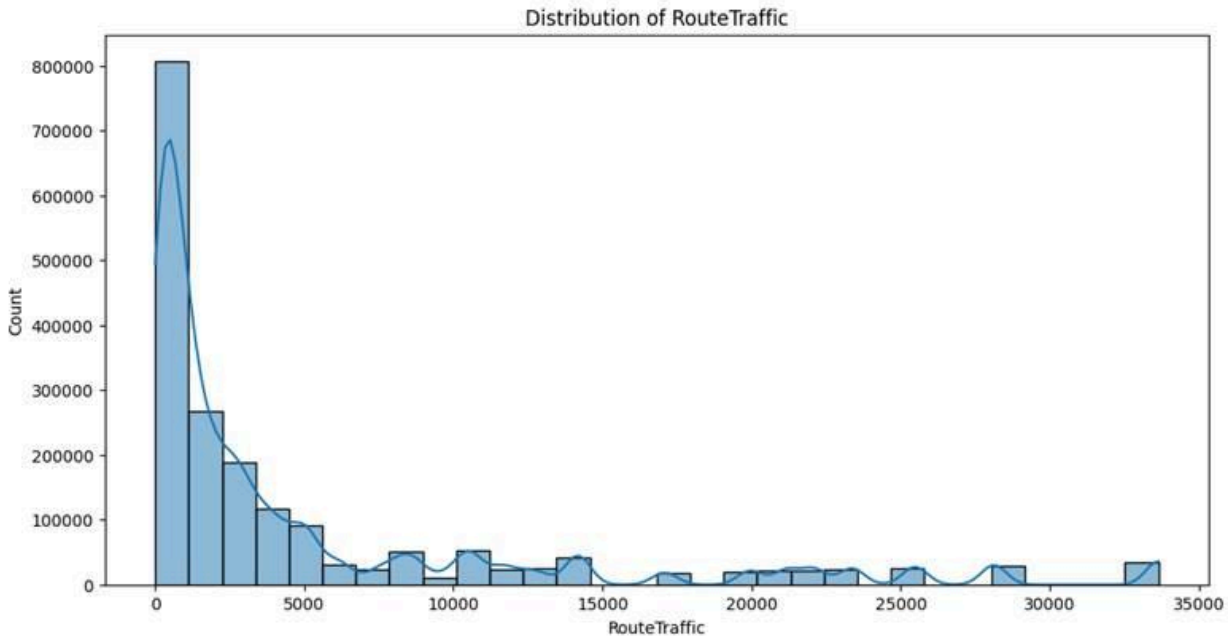
II. Exploratory Data Analysis

1. Distribution of Average Airline by age



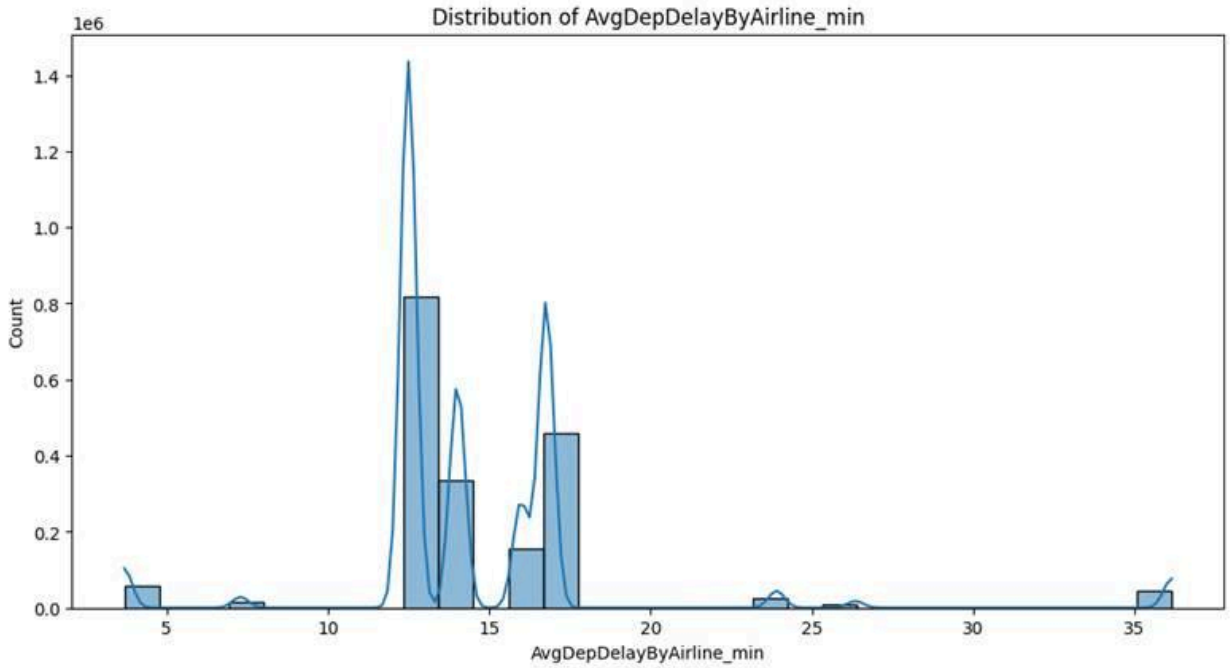
The distribution of average airline plane age is moderately right-skewed, with the majority of airlines operating fleets with an average age between 9 and 13 years. The density peaks around 11 years, indicating that many carriers operate aircraft that are just over a decade old. There are relatively few airlines with significantly newer (under 6 years) or much older (over 17 years) fleets, suggesting that most airlines maintain a balance between cost-efficiency and modernity. The right skew implies that while newer fleets are preferred, some airlines continue to use older aircraft—possibly for cost reasons or due to slower fleet renewal cycles.

2. **Distribution of RouteTraffic:**



The RouteTraffic distribution is highly right-skewed, with the majority of values concentrated at the lower end, between 0 and 2,000 passengers. A single sharp spike dominates the plot around the lowest traffic bin, indicating a large number of low-traffic routes. This suggests that a substantial number of air routes cater to relatively few passengers, possibly due to regional or niche travel demands. As the route traffic increases, the frequency drops off significantly, with only a small number of routes handling very high volumes of passengers—creating a classic long-tail distribution.

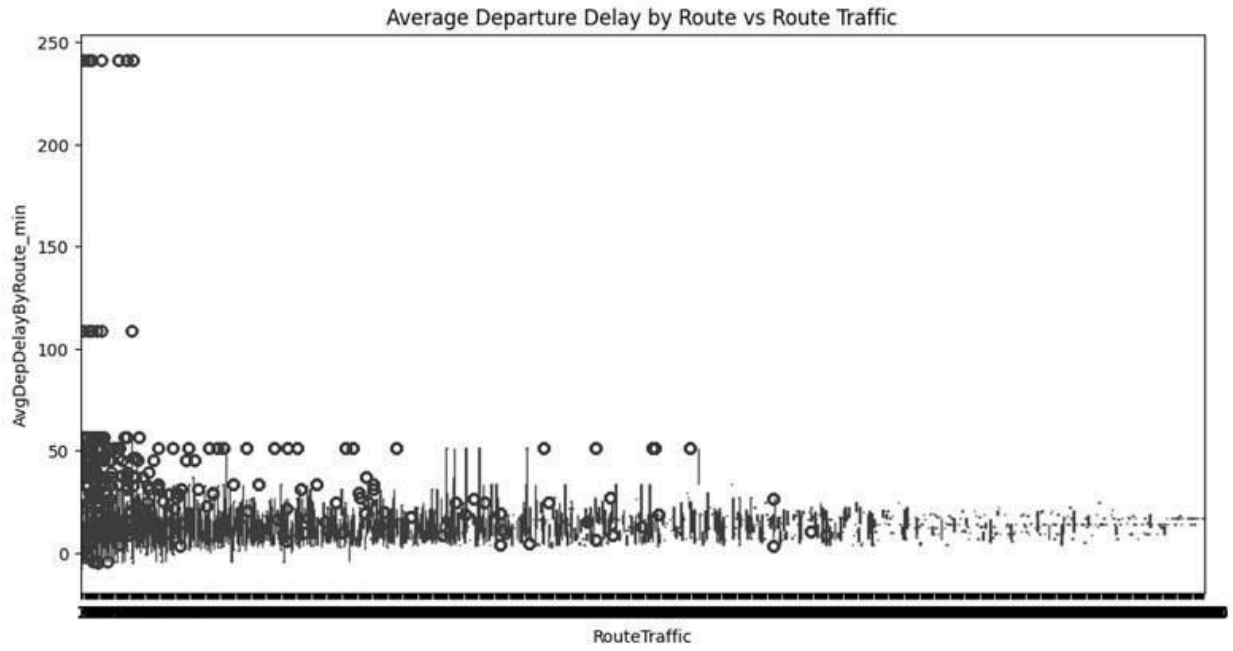
3. **Distribution of AvgDepDelayByAirline_min:**



The distribution of average departure delays by airline (in minutes) is multimodal with clear peaks

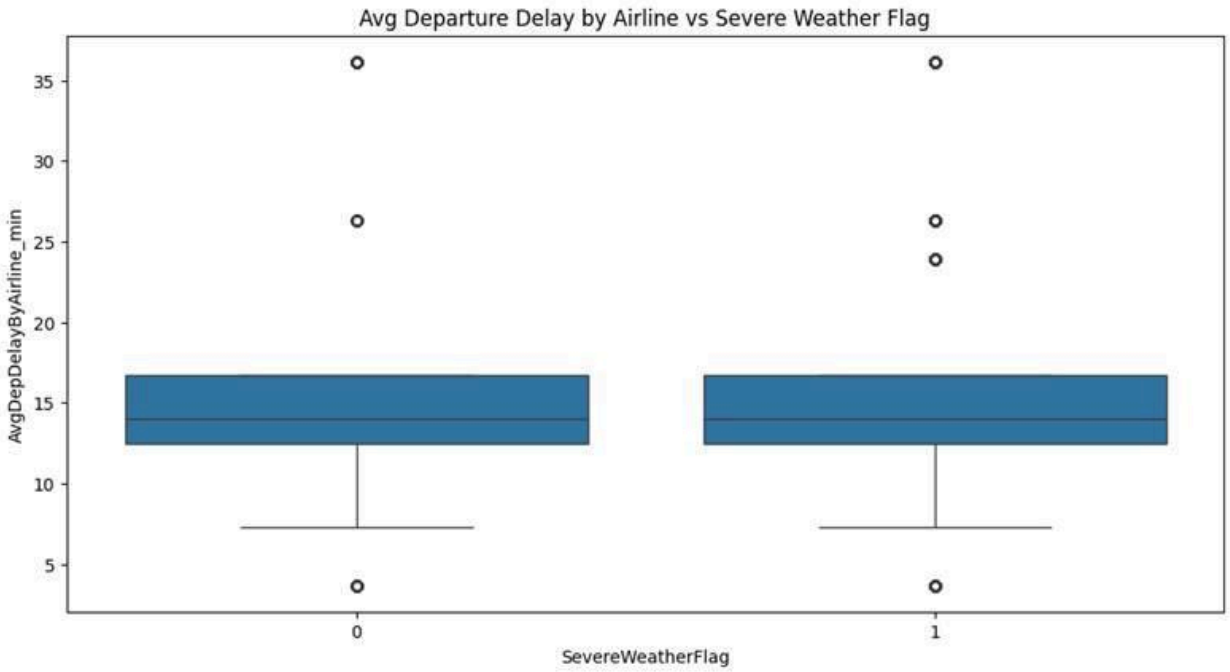
around the 13–17 minute range. This indicates that most airlines tend to cluster around this average delay range, with very few operating consistently below 10 or above 20 minutes on average. The highest density is seen just above 13 minutes, suggesting that this might be the industry norm or median level of delay for a large portion of flights. The presence of a few extreme outliers—where the average delay reaches up to 35 minutes—may represent consistently underperforming carriers or unique operational challenges

4. Average Departure Delay by Route vs Route Traffic



This scatter plot shows that most routes experience average departure delays below 50 minutes, regardless of traffic levels. However, a few low-traffic routes stand out with very high delays, exceeding 100 minutes—indicating potential outliers or route-specific issues. Overall, there is no clear linear relationship between route traffic and average delay, suggesting that traffic volume alone may not be a strong predictor of departure delays.

5. Average Departure Delay by Airline vs Severe Weather Flag



This box plot reveals that the presence of severe weather (flag = 1) does not drastically shift the average departure delay across airlines. Both weather and non-weather groups have similar medians (~14 minutes), with a slightly wider spread in delay times during severe weather. Outliers are present in both groups, but the overall impact of the severe weather flag on average delay appears minimal when viewed in isolation.

III. FEATURE SELECTION:

Target Variable: `Is_Arr_Del_Over_15_min`

The target variable selected for modeling is `Is_Arr_Del_Over_15_min`, which indicates whether a flight's arrival was delayed by more than 15 minutes.

Dropping Unnecessary Columns:

Columns removed due to irrelevance or uniqueness:

The following columns were dropped because they provide little to no value for prediction or are unique for each record (e.g., dates), making them non-informative for modeling:

Column	Justification
`FlightDate`	Each value is unique to a flight; cannot be used to predict as the same date will never come. Extracted features from it already

MAGNITUDE	Magnitude of weather is not directly useful in modeling delay likelihood.
WEATHER_TYPE	High cardinality and already reflected in other weather-derived features.
WEATHER_SOURCE	Metadata, not useful for delay prediction.
WeatherEventCountByState	Intermediate feature, replaced by SevereWeatherFlag.

Code for above:

```
# drop unnecessary columns
df.drop(columns=['FlightDate'], inplace=True)

# drop unnecasary columns, WEATHER_SOURCE is irrelevant
df.drop(columns=['MAGNITUDE', 'WEATHER_TYPE', 'WEATHER_SOURCE',
'WeatherEventCountByState'], inplace=True)
```

Dropping Columns with One-to-One Mapping:

Several columns were found to be **perfectly correlated (1:1 mapping)** with other existing fields and were dropped to reduce redundancy.

Dropped Column	Reason
STATE	Exact mapping with DestState.
DATE	Same as FlightDate.
IATA	One-to-one with Reporting_Airline.
state2	Duplicates DestState.
state1	Duplicates OriginState.
car	Identical to Reporting_Airline.
IATA_CODE_Reporting_Airline	One-to-one with Reporting_Airline.

Code Example:

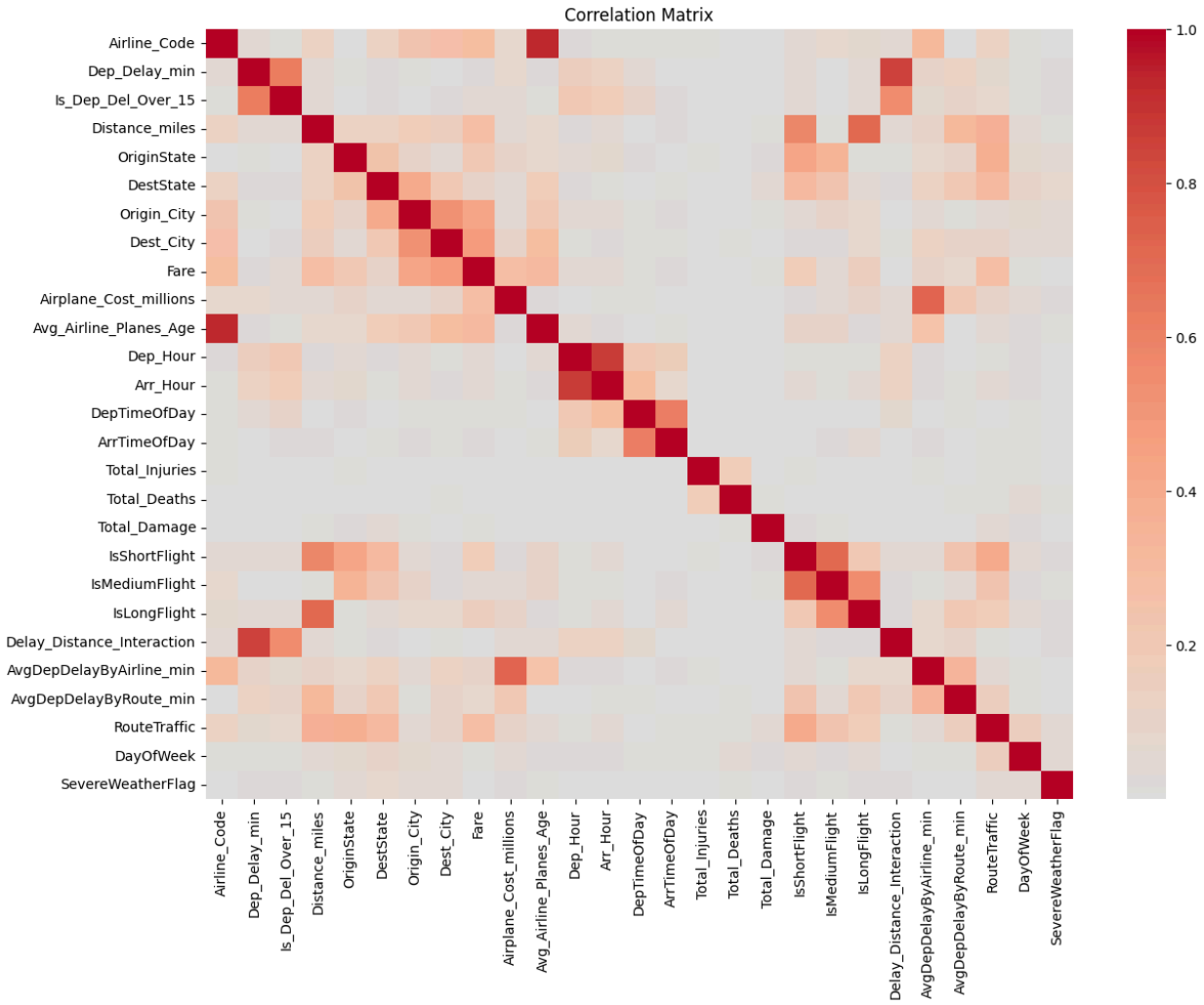
```
# Check if two columns have one-to-one mapping and drop if true
airline_to_car = df.groupby('Reporting_Airline')['Reporting_Airline'].nunique()
car_to_airline = df.groupby('car')['Reporting_Airline'].nunique()

is_one_to_one = airline_to_car.max() == 1 and car_to_airline.max() == 1

if is_one_to_one:
    df.drop(columns=['car'], inplace=True)
```

Correlation Analysis

We performed correlation analysis on numeric features and removed **highly correlated (r > 0.85)** variables to prevent multicollinearity and simplify the model.



Highly Correlated Feature Pairs:

Route <-> OriginState: 0.99
Distance_miles <-> Scheduled_Elapsed_Time_min: 0.95
Distance_miles <-> AirTime_min: 0.94
AirTime_min <-> Scheduled_Elapsed_Time_min: 0.94
Avg_Airline_Planes_Age <-> Airline_Code: 0.93
Arr_Hour <-> Dep_Hour: 0.87
Delay_Distance_Interaction <-> Dep_Delay_min: 0.85

Columns Dropped Based on Correlation:

Dropped Column	Justification
`Route`	Redundant; encoded implicitly through `OriginState` and `DestState`.
`Scheduled_Elapsed_Time_min`	Derived from distance; `Distance_miles` is more fundamental.
`AirTime_min`	Includes real-time adjustments, not available pre-flight.

Code:

```
X_encoded.drop(columns=['Route'], inplace=True)  # Redundant with OriginState + DestState
X_encoded.drop(columns=['Scheduled_Elapsed_Time_min'], inplace=True)  # Correlated with distance
X_encoded.drop(columns=['AirTime_min'], inplace=True)  # Real-time value, not known beforehand
```

IV. Data Modeling

Standardization and PCA

To ensure all features contribute equally and improve model performance, we applied **StandardScaler** to standardize the numeric features (mean = 0, standard deviation = 1).

Next, **Principal Component Analysis (PCA)** was applied to reduce dimensionality while preserving most of the variance. We retained components that explained **75% of the variance**, resulting in **12 principal components**.

Code:

```
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X_encoded)

pca = PCA(n_components=0.75, random_state=42)

X_pca = pca.fit_transform(X_scaled)
```

Train-Test Split

We split the data into **80% training and 20% testing sets**, using stratification to maintain the target class distribution.

Code:

```
X_train, X_test, y_train, y_test = train_test_split(

    X_pca, y, test_size=0.4, random_state=42, stratify=y

)
```

Output: *Train shape: (1,150,613 × 12), Test shape: (767,076 × 12)*

Model Selection

We evaluated **three classification models**, chosen for their balance between simplicity, interpretability, and power:

Model	Justification
Logistic Regression	Lightweight, interpretable baseline
Random Forest	Robust, ensemble-based, good generalization
XGBoost	Fast, scalable gradient boosting for accuracy

Model Training & Evaluation

Each model was trained on the training set and evaluated on the test set using accuracy, F1-score, and ROC AUC.

---

1. **Logistic RegressioN**
   Accuracy: 91.04%
   F1-score (class 1 - delayed): 0.8237
   Performs reasonably well but underpredicts delays due to its linear nature.

2. **Random Forest**

   Accuracy: 99.77%

   F1-score (class 1 - delayed): 0.9957

   Exceptional performance — nearly perfect classification. Slight overfitting possible, but overall very effective.

3. **XGBoost**
   Accuracy: 96.47%
   F1-score (class 1 - delayed): 0.9338
   Excellent balance between speed and performance. Highly generalizable and dashboard-friendly.

Code:

```
# ----- F1-Scores for class 1.0 -----
f1_lr = f1_score(y_test, y_pred_lr, pos_label=1)
f1_rf = f1_score(y_test, y_pred_rf, pos_label=1)
f1_xgb = f1_score(y_test, y_pred_xgb, pos_label=1)
```

Confusion matrix:

Logistic Regression

	Predicted: 0 (On-Time)	Predicted: 1 (Delayed)
Actual: 0	537,804	20,909
Actual: 1	47,808	160,555

- False Negatives (missed delays): 47,808
- False Positives (falsely predicted delays): 20,909

Random Forest

	Predicted: 0 (On-Time)	Predicted: 1 (Delayed)
Actual: 0	558,074	639
Actual: 1	1,160	207,203

- False Negatives: 1,160
- False Positives: 639

XGBoost

	Predicted: 0 (On-Time)	Predicted: 1 (Delayed)
Actual: 0	549,185	9,528
Actual: 1	17,543	190,820

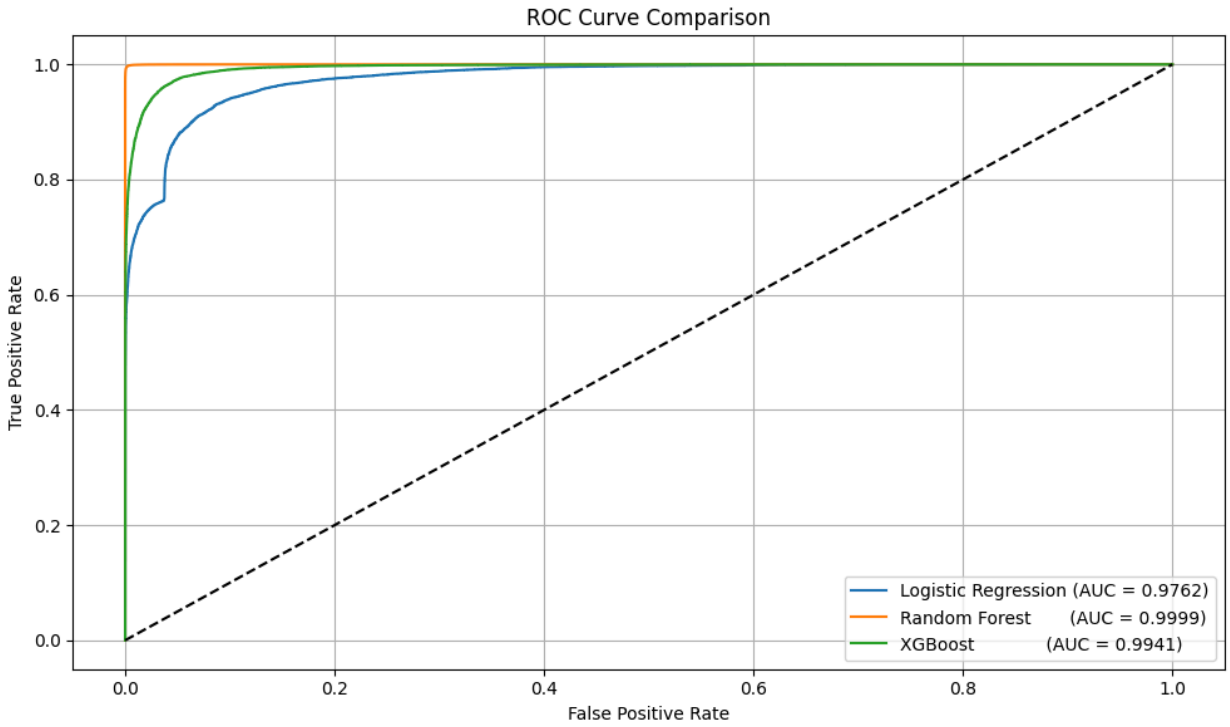
- False Negatives: 17,543
- False Positives: 9,528

Logistic Regression: High number of missed delays (false negatives), indicating underprediction of delayed flights.

Random Forest: Near-perfect classification with minimal false positives and false negatives.

XGBoost: Excellent performance with a good balance between detecting delays and avoiding false alarms.

ROC Curve:



ROC Curve Analysis:

All three models show strong ROC curves, with Random Forest achieving near-perfect separation between classes (AUC = 0.9999). XGBoost is close behind. Logistic Regression lags slightly, especially at lower false positive rates.

Code:

```
# ----- ROC Curve Plot -----
# Get prediction probabilities
y_prob_lr = lr.predict_proba(X_test)[:, 1]
y_prob_rf = rf.predict_proba(X_test)[:, 1]
y_prob_xgb = xgb.predict_proba(X_test)[:, 1]

# Compute ROC curve and AUC for each
fpr_lr, tpr_lr, _ = roc_curve(y_test, y_prob_lr)
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_prob_rf)
fpr_xgb, tpr_xgb, _ = roc_curve(y_test, y_prob_xgb)

auc_lr = auc(fpr_lr, tpr_lr)
auc_rf = auc(fpr_rf, tpr_rf)
auc_xgb = auc(fpr_xgb, tpr_xgb)

# Plot all ROC curves
plt.figure(figsize=(10, 6))
plt.plot(fpr_lr, tpr_lr, label=f"Logistic Regression (AUC = {auc_lr:.4f})")
plt.plot(fpr_rf, tpr_rf, label=f"Random Forest       (AUC = {auc_rf:.4f})")
plt.plot(fpr_xgb, tpr_xgb, label=f"XGBoost             (AUC = {auc_xgb:.4f})")

plt.plot([0, 1], [0, 1], 'k--')  # baseline
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Summary & Recommendation

Logistic Regression:
Pros: *Fast to train, interpretable, simple baseline.*
Cons: *Struggles to capture complex, nonlinear patterns in the data, resulting in* ***high false negatives*** *— i.e., many delayed flights go undetected.*

Use case: *Good for quick prototyping or environments where explainability is more important than accuracy.*

Random Forest:
Pros: *Outstanding performance across all metrics — especially* ***F1-score (0.9957)*** *and* ***AUC (0.9999)****. Very low false positives and false negatives.*
Cons: *May slightly overfit due to its ensemble depth; larger memory footprint.*
Use case: *Ideal for internal tools, backend services, or where predictive performance is the top priority.*

XGBoost:
Pros: *Strikes an excellent balance between performance and generalization. High* ***F1-score (0.9338)*** *and* ***AUC (0.9941)****, with robust delay detection capabilities and faster inference than Random Forest.*
Cons: *Slightly lower recall than Random Forest, though still very strong.*
Use case: *Best suited for* ***real-time dashboards****, APIs, and production environments due to its speed and stability.*

CONCLUSION

When building the dashboard in the next milestone, we will consider the following:

-> If **maximum accuracy** is required, go with **Random Forest**.

-> If there is a need for a **fast, scalable model for deployment**, choose **XGBoost**.
Use **Logistic Regression** only when speed or interpretability is critical, and slight accuracy loss is acceptable.

-> Each model performs well, but **Random Forest and XGBoost stand out** — making them the top candidates for real-world implementation.