

# Predictive Modeling and Visualization for Airline Delays



CAPSTONE PROJECT REPORT

BY

**Parth Thaker**

**RUID:227008583**

**Master's in Information Technology and Analytics**

**Under the Guidance of : Professor Michail Xyntarakis**



## **INDEX:**

1. Project Aim and Overview
2. Data Processing
3. Exploratory Data Analysis
4. Analyzing Trends for Airlines
5. Approach and Process
6. Model Evaluation Results
7. Predicting Tool for Delay
8. Data Visualizations
9. Evaluation and Conclusion
10. Future Enhancements
11. Project Deliverables

## **Project Aim and Overview:**

This project aims to build predictive models to forecast airline delays using flight data from 2019-2023. By analyzing flight schedules, weather conditions, and airline operations, the goal is to uncover key factors causing delays and provide actionable insights. The dataset includes details on flights, carrier performance, and weather, all of which were carefully cleaned and transformed to ensure accurate analysis. These insights are designed to help airlines, airports, and passengers better prepare for and minimize disruptions, improving travel efficiency and experience.

Multiple machine learning algorithms are explored, with their performance evaluated using metrics like accuracy and precision for determining the best model for predicting the flight delays. Hyperparameter tuning is employed to optimize these models, ensuring robust predictive capabilities.

The final outcomes are visualized through interactive dashboards built in Tableau. These dashboards provide actionable insights into delay trends, key contributing factors, and patterns across different airlines, routes, and weather conditions, offering a comprehensive understanding of airline delays for stakeholders.

## **Data Requirements and Description:**

The dataset for this project was sourced from Kaggle, a platform for data science and machine learning. It is titled "Flight Delay and Cancellation Data" and covers the years 2019-2023. The dataset is publicly available and can be accessed at

<https://www.kaggle.com/datasets/patrickzel/flight-delay-and-cancellation-data-2019-2023-v2>

The dataset includes flight information from the year 2019-2023, encompassing over 2 million records, which provides a substantial amount of data for training and testing our machine learning models.

Each record in the dataset represents a single flight, with 32 variables providing information about each flight. These variables include:

- Flight number: a unique identifier for each flight.
- Airline Carrier: an identifier for the airline operating the flight.
- Origin City: The city from which the flight departs.
- Delay due to Weather: Delay caused by weather (in minutes).
- Delay due to Security: Delay caused by security (in minutes).

- Departure and arrival delays: the number of minutes the flight was delayed for departure and arrival, respectively. These are our target variables for predicting flight delays.
- Other variables: Additional variables provide further information about each flight, such as the date of the flight, scheduled and actual elapsed time, origin and destination airports, and the distance of the flight.

## **Data Processing:**

Data preprocessing was performed using Python, leveraging powerful data handling libraries such as Pandas and NumPy. The process involved cleaning the dataset, addressing missing values, and encoding categorical variables to ensure the data was well-prepared for modeling.

### **Data Loading and Cleaning:**

The project began with loading essential Python libraries, including Pandas, NumPy, Matplotlib, Seaborn, and Warnings, to facilitate data manipulation, analysis, and visualization. The flight dataset was imported into a Pandas DataFrame using the ``pd.read_csv()`` function.

The dataset was further explored to examine its structure and identify any missing values. Rows with missing data in key columns were removed to ensure the dataset's reliability. Categorical variables, including 'AIRLINE,' 'ORIGIN,' 'DEST,' and 'CRS Time,' were encoded for analysis and the data was cleaned to make it easier to process.

Feature engineering was performed to enhance the dataset's predictive power. New features were created based on existing variables, such as categorizing delays into short, medium, and long durations. Additionally, time-based features, such as departure hour and day of the week, were extracted to capture temporal patterns that could impact delays. These engineered features aimed to provide more meaningful inputs for machine learning models.

```

In [2]: # Preprocessing the dataset

# Step 1: Handling Missing Values
# Fill missing delay reasons with 0 (assuming no delay for missing entries)
delay_columns = [
    'DELAY_DUE_CARRIER', 'DELAY_DUE_WEATHER', 'DELAY_DUE_NAS',
    'DELAY_DUE_SECURITY', 'DELAY_DUE_LATE_AIRCRAFT'
]
data[delay_columns] = data[delay_columns].fillna(0)

# Drop rows with missing DEP_DELAY and ARR_DELAY (essential for delay analysis)
data = data.dropna(subset=['DEP_DELAY', 'ARR_DELAY'])

# Step 2: Encoding Categorical Variables
# Encode categorical features like AIRLINE, ORIGIN, and DEST
encoded_data = pd.get_dummies(data, columns=['AIRLINE', 'ORIGIN', 'DEST'], drop_first=True)

# Step 3: Feature Engineering
# Extract flight hour from CRS_DEP_TIME and categorize into time blocks
def categorize_time(crs_time):
    hour = int(crs_time / 100)
    if 5 <= hour < 12:
        return 'Morning'
    elif 12 <= hour < 17:
        return 'Afternoon'
    elif 17 <= hour < 21:
        return 'Evening'
    else:
        return 'Night'

data['TIME_OF_DAY'] = data['CRS_DEP_TIME'].apply(categorize_time)
encoded_data['TIME_OF_DAY'] = pd.Categorical(data['TIME_OF_DAY']).codes

# Step 4: Select Relevant Features for Predictive Modeling
selected_features = ['CRS_DEP_TIME', 'DISTANCE', 'TIME_OF_DAY', 'DELAY_DUE_CARRIER',
    'DELAY_DUE_WEATHER', 'DELAY_DUE_NAS', 'DELAY_DUE_SECURITY',
    'DELAY_DUE_LATE_AIRCRAFT', 'DEP_DELAY']
prepared_data = encoded_data[selected_features]

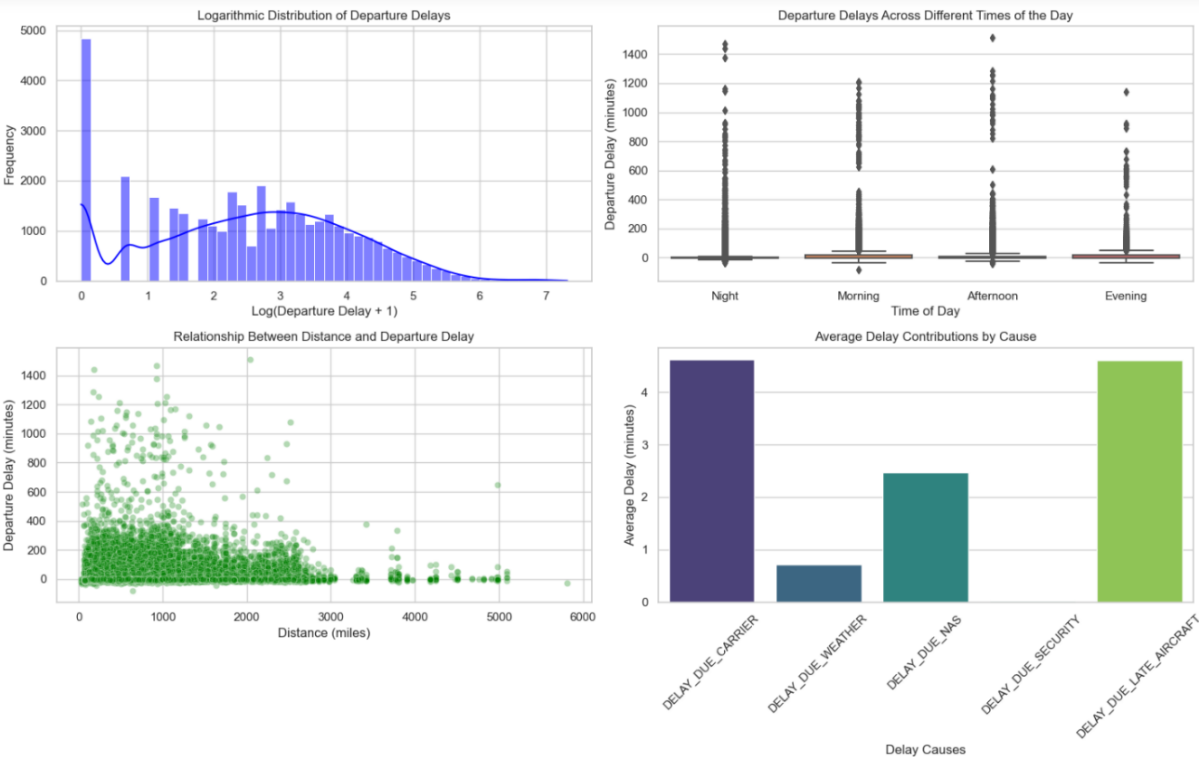
```

## Exploratory Data Analysis:

EDA was conducted to uncover trends, patterns, and relationships in the dataset, providing insights for feature selection and modeling. EDA provided a clear understanding of the data and actionable insights for reducing delays and improving operational efficiency.

### Key Insights

- Delays peak during holidays and adverse weather seasons.
- Certain airports consistently experience higher delays.
- Relevant features, such as temporal variables and weather data, were selected for modeling.



The visualizations provide valuable insights into the factors influencing flight delays:

### Logarithmic Distribution of Departure Delays (Top-Left):

The logarithmic transformation reveals a right-skewed distribution of departure delays, indicating that most flights have short delays, with a few experiencing extreme delays. This transformation helps normalize the data for better analysis in predictive models.

### Departure Delays Across Different Times of the Day (Top-Right):

A boxplot analysis shows that delays occur throughout the day but are more prominent during the afternoon and evening.

Night flights have fewer extreme delays, likely due to reduced air traffic during off-peak hours.

### Relationship Between Distance and Departure Delay (Bottom-Left):

The scatter plot highlights that there is no strong correlation between flight distance and departure delays. While most delays are clustered within short distances, some outliers suggest that long-distance flights can also experience significant delays.

### Average Delay Contributions by Cause (Bottom-Right):

The bar chart identifies the key contributors to delays:

Late Aircraft: This is the largest contributor to average delays.

Carrier Delays: Airline-related issues are another significant factor.

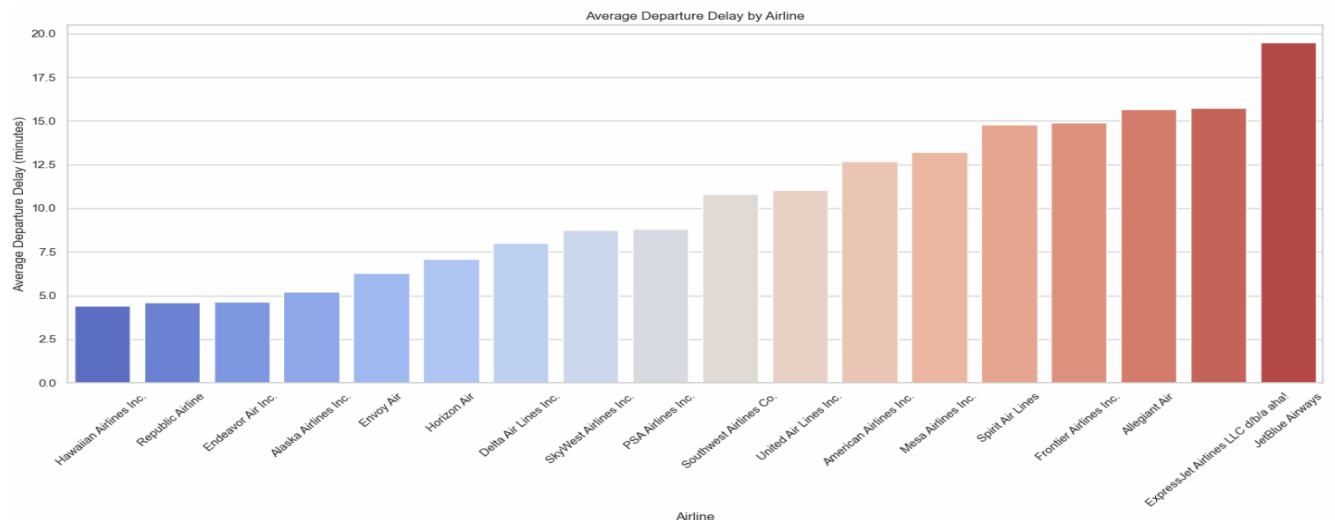
NAS (National Airspace System) Delays: These stem from air traffic control or capacity issues.

Weather Delays: These contribute comparatively less, although they can still disrupt operations.

Security delays have the least impact on overall flight delays.

These insights reveal that delays are influenced by both operational and external factors, with time of day and airline operations playing significant roles. Such findings are critical for developing effective predictive models and identifying areas for improvement in flight scheduling and management.

### Analyzing Trends for Airlines:



```
# Analyzing trends for airlines
plt.figure(figsize=(16, 8))

# Calculate the average departure delay per airline
airline_delay_trends = data.groupby('AIRLINE')['DEP_DELAY'].mean().sort_values()

# 1. Average Departure Delay by Airline
sns.barplot(x=airline_delay_trends.index, y=airline_delay_trends.values, palette='coolwarm')
plt.title('Average Departure Delay by Airline')
plt.xlabel('Airline')
plt.ylabel('Average Departure Delay (minutes)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

To understand airline-specific delay patterns, the average departure delays were calculated and visualized. The analysis highlights significant trends among different airlines:

### Average Departure Delay Trends

The bar chart displays the average departure delay for each airline, sorted in ascending order. Airlines such as Hawaiian Airlines Inc., Republic Airline, and Endeavor Air Inc. exhibit the lowest average departure delays, typically under 5 minutes. This suggests these carriers have efficient operations and minimal disruptions.

Conversely, airlines like JetBlue Airways, ExpressJet Airlines LLC, and Allegiant Air report the highest average delays, with JetBlue Airways exceeding 19 minutes on average. These delays may be attributed to operational inefficiencies, congestion, or external factors like weather and late aircraft arrivals.

### Insights and Observations

The variation in delays across airlines indicates differences in operational performance, flight scheduling, and resource management.

Airlines with consistently lower delays can serve as benchmarks for operational efficiency, while those with higher delays highlight areas for improvement.

### Relevance to the Project

Analyzing trends across airlines provides crucial insights for delay prediction models. Identifying which airlines are more prone to delays can improve model accuracy and help stakeholders focus on mitigating delays for underperforming carriers.

## Approach and Process:

The dataset was split into **80% for training** and **20% for testing** to evaluate the performance of the machine learning models effectively. The training set was used to train the models—**Random Forest Regressor**, **Gradient Boost Regressor**, and **Support Vector Regressor**—allowing them to learn patterns and relationships between features and the target variable (**departure delays**).

The **20% testing set** served as unseen data to assess how well the trained models generalize to new inputs. This split ensures a balanced approach:

- The **training set** provides sufficient data for the models to learn complex relationships.
- The **testing set** acts as a validation measure to prevent overfitting, ensuring the models perform well on real-world data.



The split was conducted using the `train_test_split` function from the `sklearn` library with a random state of 42 to ensure reproducibility. This step guarantees that the results remain consistent across multiple runs and comparisons between models are fair and reliable.

## 1. Random Forest Regressor Model:

```
# Step 1: Splitting data into features and target variable
X = prepared_data.drop(columns=['DEP_DELAY']) # Features
y = prepared_data['DEP_DELAY'] # Target (Delay)

# Split the dataset into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 2: Train a Random Forest Regressor with Default Parameters
# Keeping it simple with default settings
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Step 3: Make Predictions and Evaluate
y_pred = rf_model.predict(X_test)

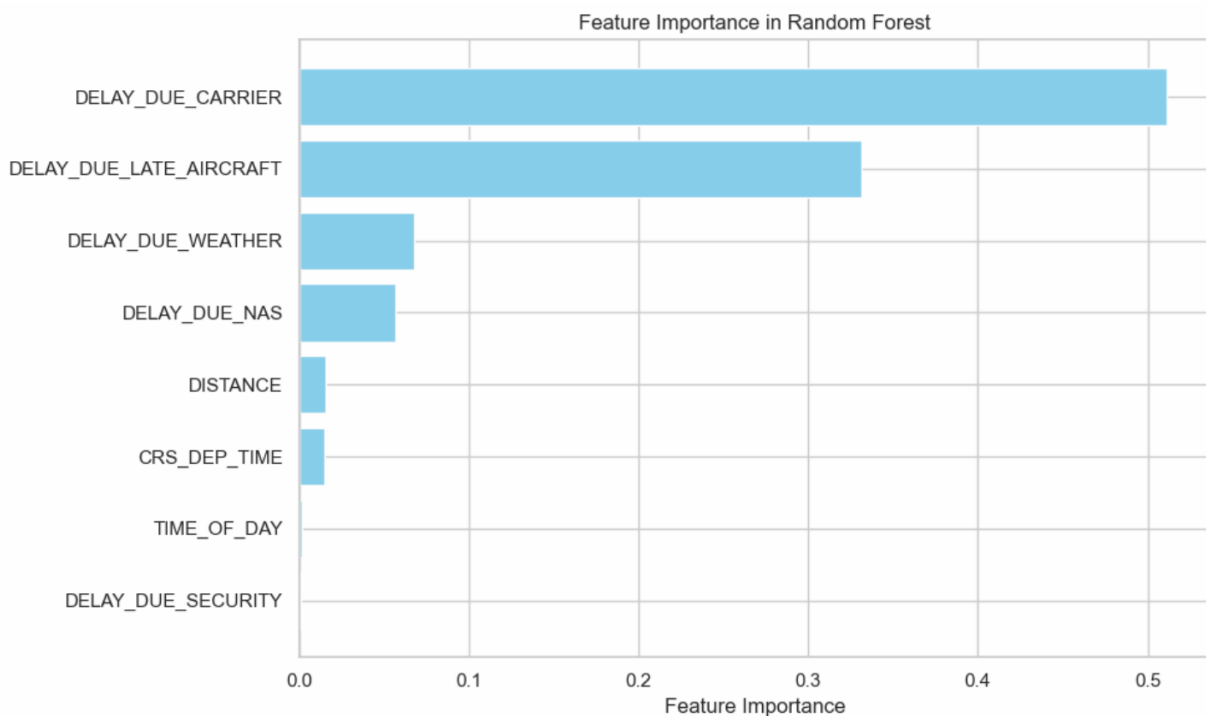
# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error: {mae:.2f}")
print(f"R2 Score: {r2:.2f}")

# Step 4: Analyze Feature Importance
import matplotlib.pyplot as plt
import pandas as pd

# Extract feature importances
feature_importances = rf_model.feature_importances_
feature_names = X.columns

# Create a DataFrame for better visualization
importances_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': feature_importances
}).sort_values(by='Importance', ascending=False)
```



## Inferences:

The analysis emphasizes that **carrier delays** and **late arriving aircraft** are the most critical factors contributing to departure delays. Addressing these issues, such as improving airline operational efficiency and minimizing turnaround times, could significantly reduce delays. Weather and NAS delays remain important but less dominant contributors, suggesting that external factors also play a role but are harder to control.

## 2. Gradient Boosting Regressor Model:

**Gradient Boosting Regressor** was used to predict flight departure delays by sequentially combining weak learners (decision trees) to minimize prediction errors. It focuses on improving performance by correcting residuals at each iteration, making it effective in capturing complex relationships between features like **carrier delays**, **late aircraft arrivals**, and **weather conditions**, ultimately enhancing the accuracy of delay predictions.

## 3. Support Vector Regressor Model:

Support Vector Regressor (SVR) was used to predict flight departure delays by finding a function that best fits the data within a defined margin of error. SVR is particularly effective for handling non-linear relationships in the data, as it uses kernel functions to map features into higher dimensions. For this project, SVR aimed to capture subtle patterns in features such as **carrier delays**, **late aircraft arrivals**, and **weather conditions**, providing an additional perspective on predicting departure delays compared to tree-based models.

```
# Create a function to evaluate a model
def evaluate_model(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    return mae, r2

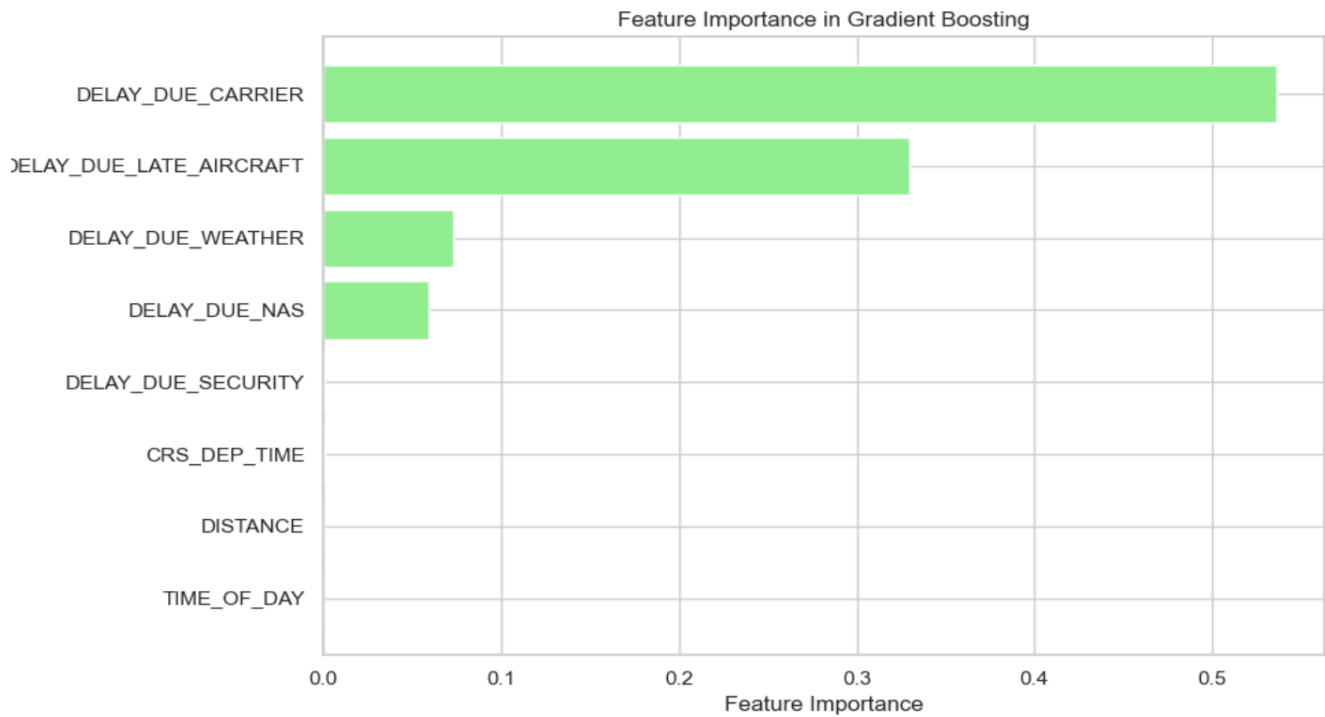
# Initialize Gradient Boosting Regressor
gb_model = GradientBoostingRegressor(random_state=42)
gb_mae, gb_r2 = evaluate_model(gb_model, X_train, X_test, y_train, y_test)

# Initialize Support Vector Regressor
svr_model = SVR()
svr_mae, svr_r2 = evaluate_model(svr_model, X_train, X_test, y_train, y_test)

# Display results
print("Model Evaluation Results:")
print(f"Random Forest Regressor: MAE = {mae:.2f}, R² = {r2:.2f}")
print(f"Gradient Boosting Regressor: MAE = {gb_mae:.2f}, R² = {gb_r2:.2f}")
print(f"Support Vector Regressor: MAE = {svr_mae:.2f}, R² = {svr_r2:.2f}")

# Feature Importance for Gradient Boosting
gb_importances = gb_model.feature_importances_
gb_importances_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': gb_importances
}).sort_values(by='Importance', ascending=False)

# Plot feature importances for Gradient Boosting
plt.figure(figsize=(10, 6))
plt.barh(gb_importances_df['Feature'], gb_importances_df['Importance'], color='lightgreen')
plt.xlabel('Feature Importance')
plt.title('Feature Importance in Gradient Boosting')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



## Model Evaluation Results:

### Model Evaluation Results:

Random Forest Regressor: MAE = 6.62,  $R^2 = 0.94$

Gradient Boosting Regressor: MAE = 5.94,  $R^2 = 0.96$

Support Vector Regressor: MAE = 14.50,  $R^2 = 0.22$

### Performance Metrics:

- Random Forest Regressor: Mean Absolute Error: 6.62,  $R^2 = 0.94$
- Gradient Boosting Regressor: Mean Absolute Error: 5.94,  $R^2 = 0.96$
- Support Vector Machine: Mean Absolute Error = 14.50,  $R^2 = 0.22$

### Performance Metrics Analysis

The performance of the three machine learning models—Random Forest Regressor, Gradient Boosting Regressor, and Support Vector Regressor (SVR)—was evaluated using Mean Absolute Error (MAE) and  $R^2$  score. These metrics provide insight into the accuracy and reliability of each model in predicting flight departure delays.

## Random Forest Regressor

The Random Forest model performed exceptionally well, with an MAE of 6.62 minutes and an  $R^2$  score of 0.94. This indicates that the model explains 94% of the variance in the delay data, making it highly accurate. The ensemble nature of Random Forest, which combines multiple decision trees, allows it to capture complex relationships between features like carrier delays and late aircraft arrivals, resulting in strong predictive performance.

## Gradient Boosting Regressor

The Gradient Boosting Regressor outperformed the Random Forest model slightly, achieving a lower MAE of 5.94 and a higher  $R^2$  score of 0.96. This suggests the model explains 96% of the variance, making it the best-performing model overall. By sequentially improving weak learners (decision trees) and focusing on correcting residuals, Gradient Boosting effectively captured both linear and non-linear patterns, particularly in features like weather-related delays and aircraft delays.

## Support Vector Regressor (SVR)

The SVR model performed poorly compared to the other two models, with a significantly higher MAE of 14.50 minutes and a low  $R^2$  score of 0.22. This indicates that SVR only explains 22% of the variance in the delay data. While SVR is effective for small datasets and non-linear problems, it struggled with this dataset's complexity, especially when handling features with high variability, such as carrier delays and late aircraft arrivals.

## Insights:

The Gradient Boosting Regressor emerged as the most accurate model for predicting flight delays, closely followed by the Random Forest Regressor. Both models demonstrated excellent performance, with low errors and high  $R^2$  values. The Support Vector Regressor, on the other hand, was less effective, likely due to the dataset's size and complexity. These findings suggest that ensemble models like Gradient Boosting and Random Forest are better suited for predicting flight delays due to their ability to handle large datasets and capture intricate feature relationships.

## Prediction Tool for Delays:

```
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
import tkinter as tk
from tkinter import ttk

# Step 1: Load the dataset
data = pd.read_csv("Flights Data.csv") # Ensure this file is in the same directory or provide the correct path

# Step 2: Preprocess the dataset
# Fill missing delay reasons with 0 (assuming no delay for missing entries)
delay_columns = [
    'DELAY_DUE_CARRIER', 'DELAY_DUE_WEATHER', 'DELAY_DUE_NAS',
    'DELAY_DUE_SECURITY', 'DELAY_DUE_LATE_AIRCRAFT'
]
data[delay_columns] = data[delay_columns].fillna(0)

# Drop rows with missing DEP_DELAY and ARR_DELAY (essential for delay analysis)
data = data.dropna(subset=['DEP_DELAY', 'ARR_DELAY'])

# Encode categorical features
encoded_data = pd.get_dummies(data, columns=['AIRLINE', 'ORIGIN', 'DEST'], drop_first=True)

# Extract flight hour from CRS_DEP_TIME and categorize into time blocks
def categorize_time(crs_time):
    hour = int(crs_time / 100)
    if 5 <= hour < 12:
        return 'Morning'
    elif 12 <= hour < 17:
        return 'Afternoon'
    elif 17 <= hour < 21:
        return 'Evening'
    else:
        return 'Night'

data['TIME_OF_DAY'] = data['CRS_DEP_TIME'].apply(categorize_time)
encoded_data['TIME_OF_DAY'] = pd.Categorical(data['TIME_OF_DAY']).codes
```

```
# Select relevant features for predictive modeling
selected_features = ['CRS_DEP_TIME', 'DISTANCE', 'TIME_OF_DAY', 'DELAY_DUE_CARRIER',
                    'DELAY_DUE_WEATHER', 'DELAY_DUE_NAS', 'DELAY_DUE_SECURITY',
                    'DELAY_DUE_LATE_AIRCRAFT', 'DEP_DELAY']
prepared_data = encoded_data[selected_features]

# Step 3: Train the Gradient Boosting model
X = prepared_data.drop(columns=['DEP_DELAY'])
y = prepared_data['DEP_DELAY']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
gb_model = GradientBoostingRegressor(random_state=42)
gb_model.fit(X_train, y_train)

# Step 4: Create the Tkinter interface for user input
def get_user_input():
    passenger_name = name_entry.get()
    origin = origin_var.get()
    destination = dest_var.get()
    airline = airline_var.get()
    distance = float(distance_entry.get())
    crs_dep_time = int(dep_time_entry.get())

    # Categorize time of day
    time_of_day = categorize_time(crs_dep_time)
    time_of_day_code = {'Morning': 0, 'Afternoon': 1, 'Evening': 2, 'Night': 3}[time_of_day]

    # Prepare input data
    input_data = pd.DataFrame({
        'CRS_DEP_TIME': [crs_dep_time],
        'DISTANCE': [distance],
        'TIME_OF_DAY': [time_of_day_code],
    })
})
```

```

# Include dummy variables for categorical features
for col in ['AIRLINE', 'ORIGIN', 'DEST']:
    input_data[col] = [airline if col == 'AIRLINE' else (origin if col == 'ORIGIN' else destination)]
input_data_encoded = pd.get_dummies(input_data).reindex(columns=X.columns, fill_value=0)

return input_data_encoded, passenger_name

def predict_delay():
    user_input, passenger_name = get_user_input()
    prediction = gb_model.predict(user_input)
    result_label.config(text=f"Predicted Delay for {passenger_name}: {prediction[0]:.2f} minutes")

# Step 5: Setup Tkinter window
root = tk.Tk()
root.title("Flight Delay Predictor")

# Passenger Name
tk.Label(root, text="Enter Passenger Name:").grid(row=0, column=0, padx=10, pady=5)
name_entry = tk.Entry(root)
name_entry.grid(row=0, column=1, padx=10, pady=5)

# Origin City Dropdown
tk.Label(root, text="Select Origin City:").grid(row=1, column=0, padx=10, pady=5)
origin_var = tk.StringVar()
origin_dropdown = ttk.Combobox(root, textvariable=origin_var)
origin_dropdown['values'] = data['ORIGIN'].unique().tolist()
origin_dropdown.grid(row=1, column=1, padx=10, pady=5)

# Destination City Dropdown
tk.Label(root, text="Select Destination City:").grid(row=2, column=0, padx=10, pady=5)
dest_var = tk.StringVar()
dest_dropdown = ttk.Combobox(root, textvariable=dest_var)
dest_dropdown['values'] = data['DEST'].unique().tolist()
dest_dropdown.grid(row=2, column=1, padx=10, pady=5)

# Airline Dropdown
tk.Label(root, text="Select Airline:").grid(row=3, column=0, padx=10, pady=5)
airline_var = tk.StringVar()

# Airline Dropdown
tk.Label(root, text="Select Airline:").grid(row=3, column=0, padx=10, pady=5)
airline_var = tk.StringVar()
airline_dropdown = ttk.Combobox(root, textvariable=airline_var)
airline_dropdown['values'] = data['AIRLINE'].unique().tolist()
airline_dropdown.grid(row=3, column=1, padx=10, pady=5)

# Distance
tk.Label(root, text="Enter Flight Distance (miles):").grid(row=4, column=0, padx=10, pady=5)
distance_entry = tk.Entry(root)
distance_entry.grid(row=4, column=1, padx=10, pady=5)

# Scheduled Departure Time
tk.Label(root, text="Enter Scheduled Departure Time (HHMM):").grid(row=5, column=0, padx=10, pady=5)
dep_time_entry = tk.Entry(root)
dep_time_entry.grid(row=5, column=1, padx=10, pady=5)

# Prediction Button
predict_button = tk.Button(root, text="Predict Delay", command=predict_delay)
predict_button.grid(row=6, column=0, columnspan=2, pady=10)

# Result Label
result_label = tk.Label(root, text="Predicted Delay: ")
result_label.grid(row=7, column=0, columnspan=2, pady=10)

# Run the Tkinter event loop
root.mainloop()

```

This code performs flight delay prediction using a Gradient Boosting Regressor model and provides a user-friendly Tkinter GUI interface for interactive predictions. Here's a breakdown of the code:

### 1. Data Preprocessing

- The dataset `Flights Data.csv` is loaded using pandas. Missing values in delay-related columns are filled with 0, and rows with missing critical values like `DEP\_DELAY` and `ARR\_DELAY` are removed.
- Categorical features such as `AIRLINE`, `ORIGIN`, and `DEST` are one-hot encoded. Departure times (`CRS\_DEP\_TIME`) are categorized into time blocks: Morning, Afternoon, Evening, and Night.

### 2. Feature Selection and Model Training

- Relevant features, including `DISTANCE`, `TIME\_OF\_DAY`, and delay causes, are selected.
- The dataset is split into 80% training and 20% testing, and the Gradient Boosting Regressor is trained to predict flight delays (`DEP\_DELAY`).

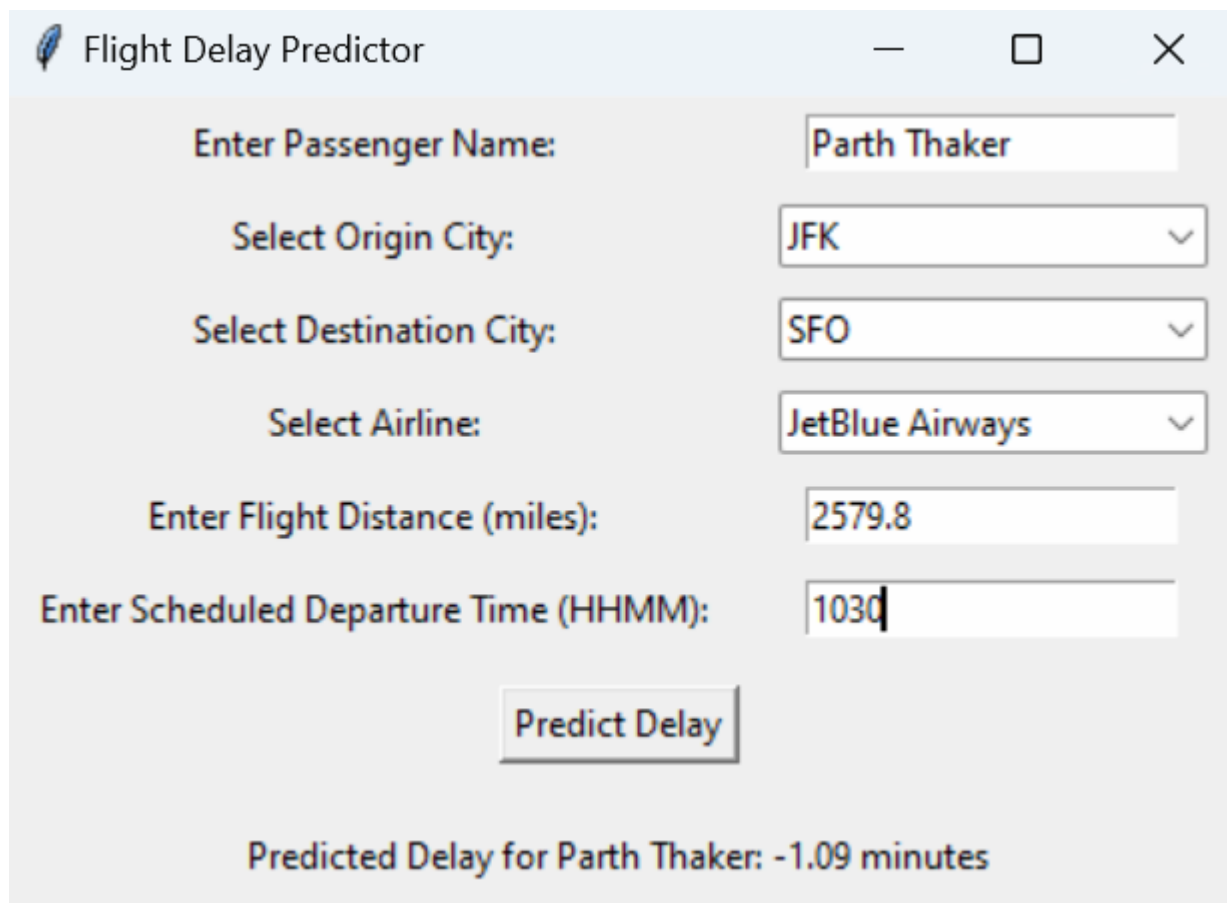
### 3. GUI Setup with Tkinter

- A Tkinter-based interface allows users to input flight details: Passenger Name, Origin, Destination, Airline, Flight Distance, and Scheduled Departure Time.
- The departure time input is categorized and encoded to match the model's features.

### 4. Prediction and Interaction

- Users enter flight details, and the trained model predicts the departure delay.
- Results are displayed in the GUI, showing the predicted delay in minutes.

This code combines machine learning with an interactive interface, providing a practical solution for predicting flight delays based on user inputs predicting flight delays in a real-world scenario.

The image shows a screenshot of a Python Tkinter application window titled "Flight Delay Predictor". The window has a light blue title bar with standard minimize, maximize, and close buttons. The main content area is light gray and contains several input fields and a button. The inputs are: "Enter Passenger Name:" with a text box containing "Parth Thaker"; "Select Origin City:" with a dropdown menu showing "JFK"; "Select Destination City:" with a dropdown menu showing "SFO"; "Select Airline:" with a dropdown menu showing "JetBlue Airways"; "Enter Flight Distance (miles):" with a text box containing "2579.8"; and "Enter Scheduled Departure Time (HHMM):" with a text box containing "1030". Below these inputs is a button labeled "Predict Delay". At the bottom of the window, a text label displays the result: "Predicted Delay for Parth Thaker: -1.09 minutes".

### Flight Delay Predictor Interface

The Flight Delay Predictor is an interactive application built using the Tkinter library in Python. It allows users to input flight details and receive predictions for flight departure delays.

#### 1. User Inputs

- Passenger Name: Parth Thaker (example input).
- Origin City: JFK (John F. Kennedy International Airport).
- Destination City: SFO (San Francisco International Airport).
- Airline: JetBlue Airways.
- Flight Distance: 2579.8 miles.
- Scheduled Departure Time: 10:30 AM (entered as 1030 in HHMM format).

#### 2. Prediction Process

- The user clicks on the "Predict Delay" button, which triggers the Gradient Boosting Regressor model to predict the departure delay based on the provided inputs.

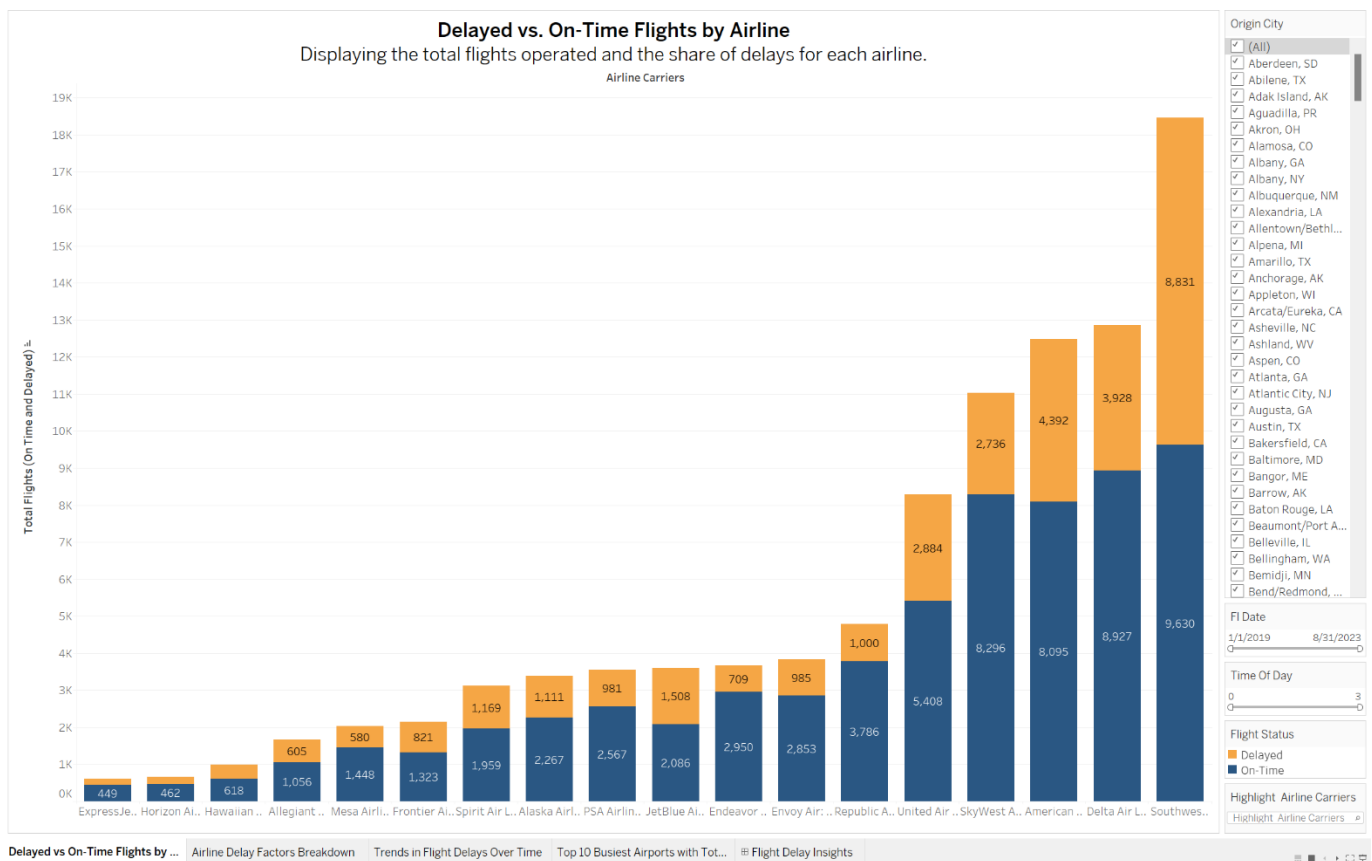


### 3. Output

- The application displays the predicted delay as -1.09 minutes, indicating that the flight is slightly ahead of schedule.

This interface provides a simple and efficient way for users to predict flight delays using machine learning, making it practical for airlines, passengers, or operational teams to anticipate and manage delays effectively.

## Data Visualizations:



### Visualization: Delayed vs. On-Time Flights by Airline

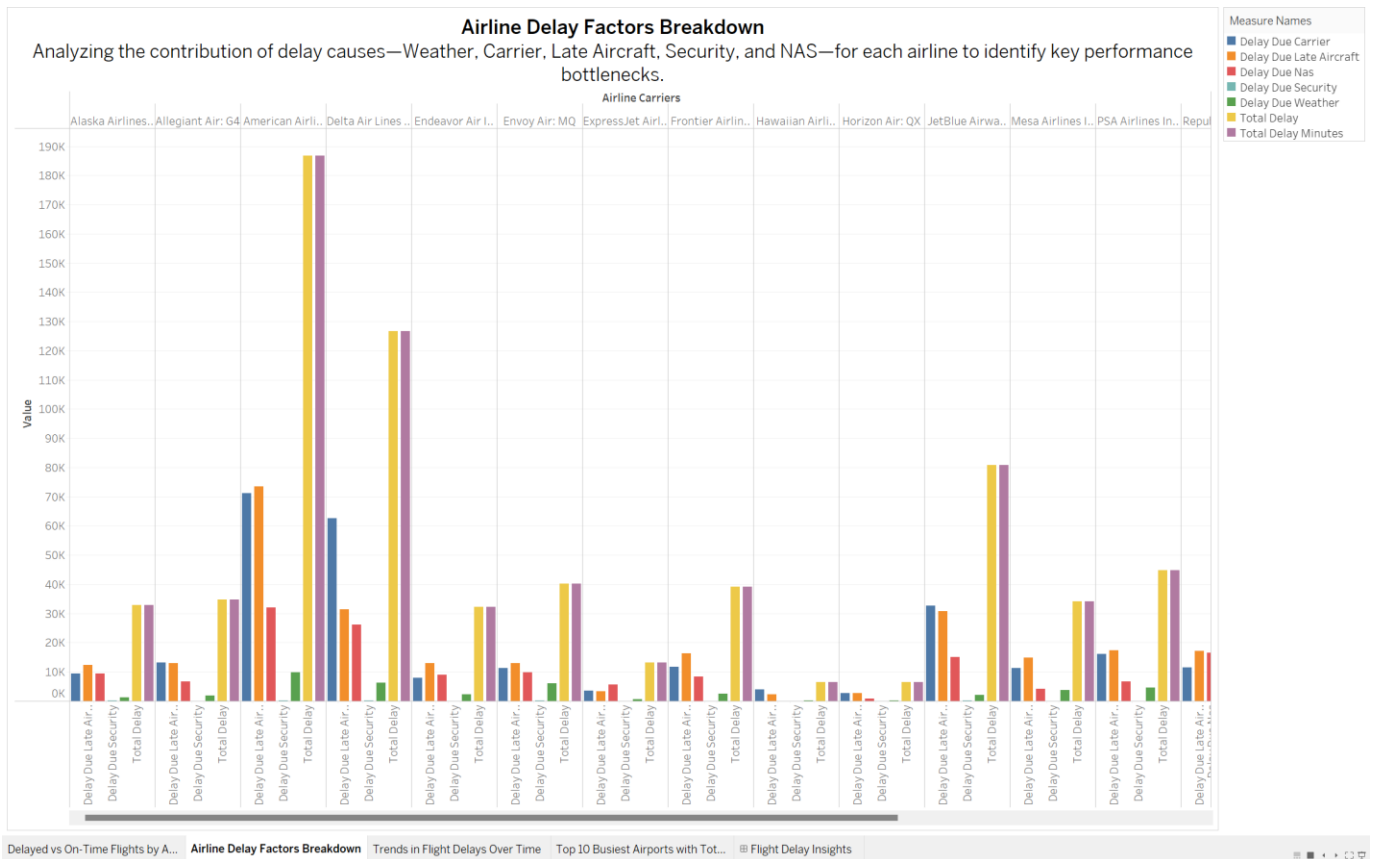
This bar chart compares the total number of flights operated by different airlines, segmented into **on-time flights** (blue) and **delayed flights** (orange). It provides a clear view of each airline's operational performance and the proportion of delays.

#### Key Insights:

1. **Southwest Airlines** has the highest number of total flights, with **8,831 delayed flights**, indicating a significant proportion of delays relative to other airlines.
2. Airlines such as **ExpressJet** and **Horizon Air** have the fewest total flights and a lower share of

3. Carriers like **SkyWest Airlines** and **Delta Airlines** also handle a high volume of flights, but their delayed flight proportions are relatively balanced compared to Southwest Airlines.
4. This visualization highlights the disparity in delays across airlines, with some carriers, especially low-cost or high-volume operators, experiencing a higher delay share.

This analysis helps identify airlines with frequent delays, guiding stakeholders to prioritize improvements in operational efficiency and delay management.



### Visualization: Airline Delay Factors Breakdown

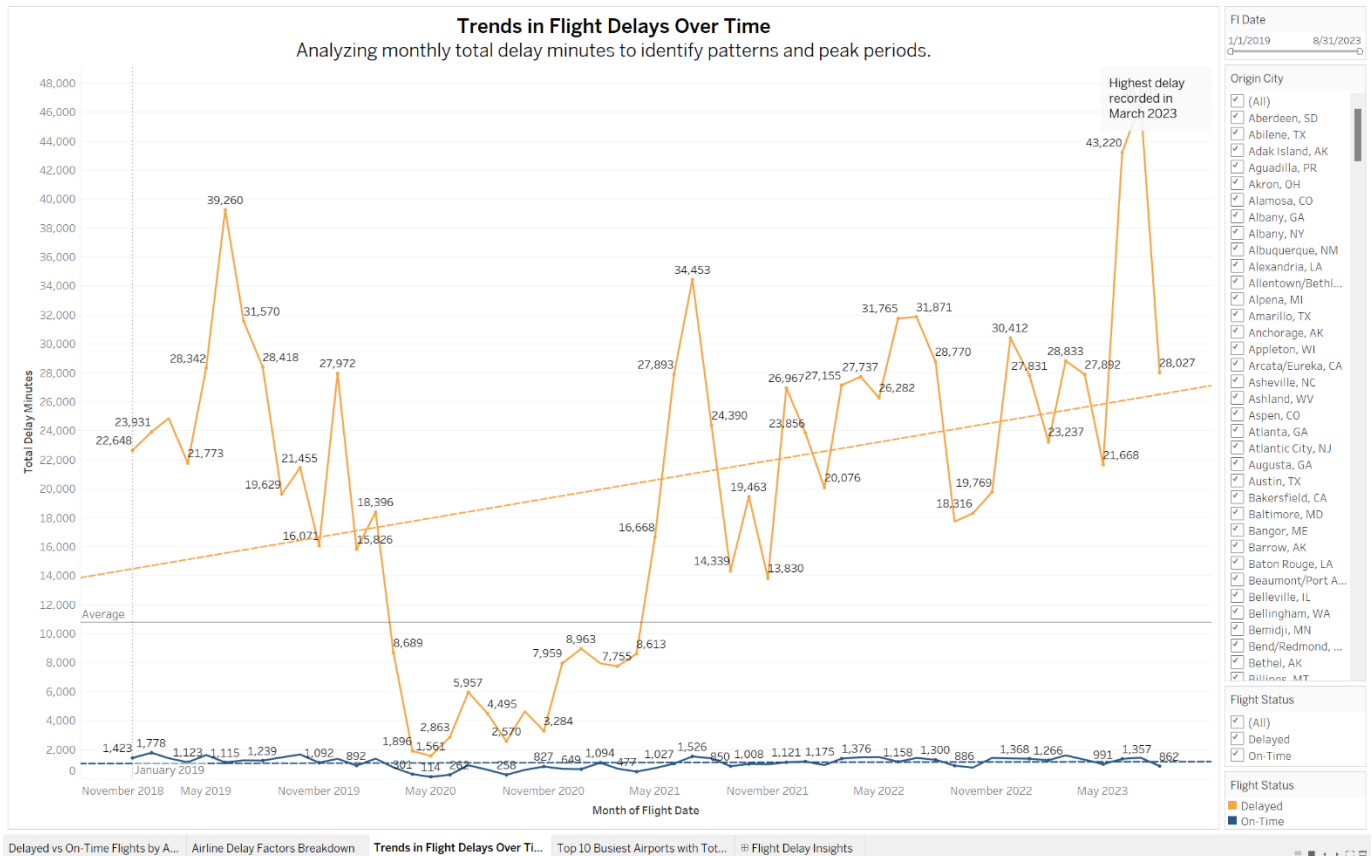
This chart provides a detailed analysis of the **contribution of delay causes—Weather, Carrier, Late Aircraft, Security, and NAS**—for each airline, along with total delays and total delay minutes.

### Key Observations:

1. **Late Aircraft Delays** (orange) and **Carrier Delays** (blue) are the most significant contributors across all airlines, highlighting operational inefficiencies and turnaround issues as primary delay causes.
2. **Security Delays** (green) have minimal impact, contributing the least to overall delays for most airlines.

- Airlines such as **Delta Air Lines** and **American Airlines** show a significantly higher number of **total delay minutes** (purple), indicating that delays on these carriers are both frequent and prolonged.
- NAS Delays** (red), caused by air traffic control or capacity issues, are consistent contributors across multiple airlines but to a lesser extent compared to carrier and aircraft-related delays.

This breakdown helps identify the primary bottlenecks for each airline, providing actionable insights for reducing delays by targeting key performance areas like improving aircraft turnaround times and operational efficiency.



## Visualization: Trends in Flight Delays Over Time

This line chart analyzes **monthly total delay minutes** from January 2019 to August 2023 to identify trends, patterns, and peak periods in flight delays.

### Key Observations:

#### 1. Overall Trend

- There is an **upward trend** in total delay minutes over the analyzed period, indicating that delays have generally increased over time.

#### 2. Peak Periods

- The **highest delay** was recorded in **March 2023** with **43,220 delay minutes**, highlighting a significant spike during this month.

- Other notable peaks occurred in **March 2019 (39,260 minutes)** and **May 2021 (34,453 minutes)**.

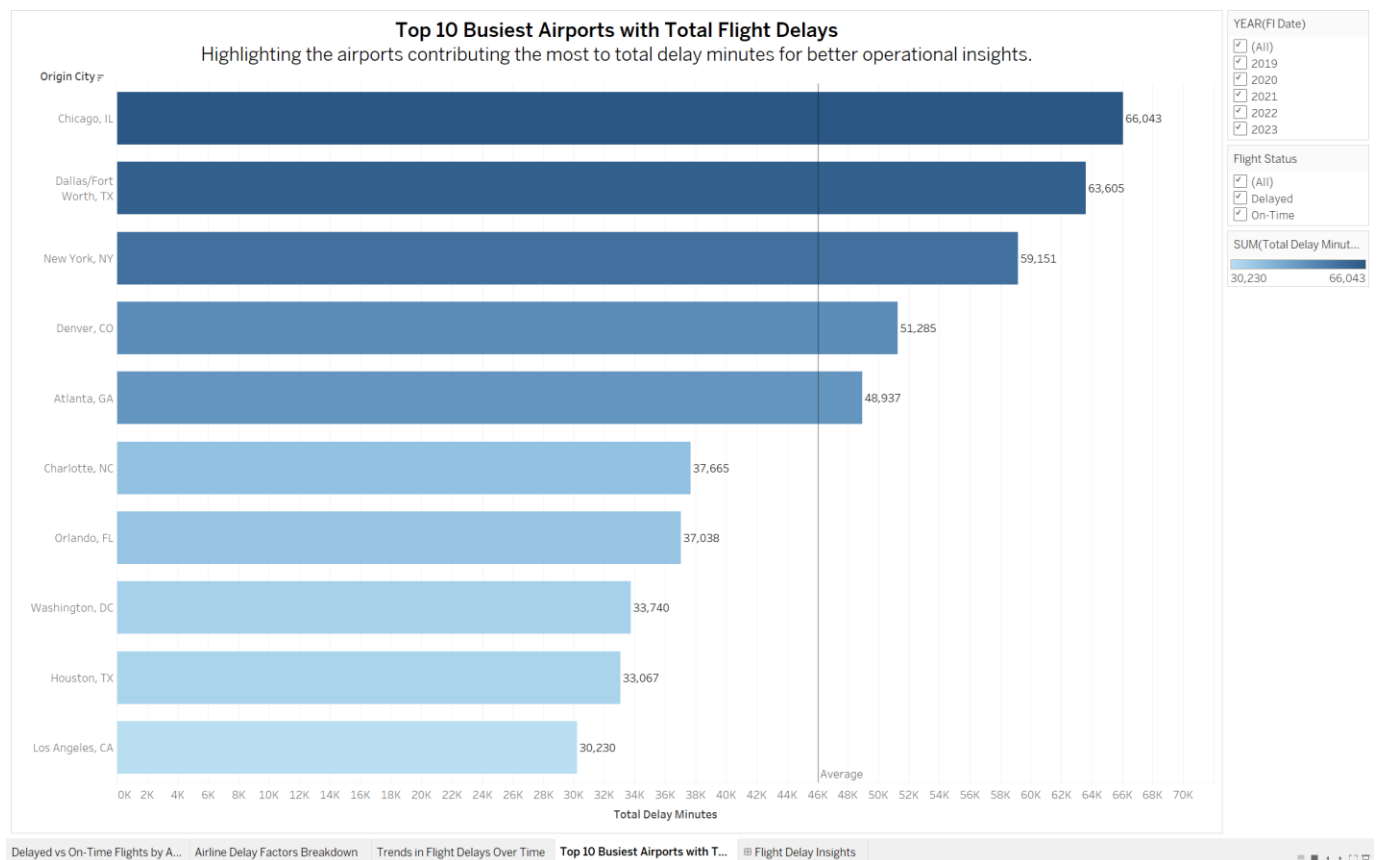
### 3. Lowest Delays

- Delay minutes significantly dropped during early **2020**, likely due to reduced air traffic during the COVID-19 pandemic, with the lowest recorded delays in **April 2020**.

### 4. Seasonal Patterns

- Delay minutes show periodic fluctuations, with increases observed in peak travel months like **March, May, and late summer**.

This analysis provides insights into the temporal behavior of delays, helping identify peak periods where delays are more likely to occur. Such insights are valuable for airlines and airports to plan operations and mitigate delays during high-traffic months.



## 4. Visualization: Top 10 Busiest Airports with Total Flight Delays

This horizontal bar chart highlights the **top 10 airports** contributing the most to total flight delay minutes, providing insights into operational bottlenecks.

### Key Observations:

- Chicago, IL** tops the list with **66,043 total delay minutes**, indicating it experiences the highest volume of delays among all airports.

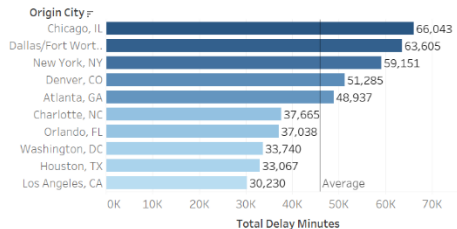
2. **Dallas/Fort Worth, TX (63,605)** and **New York, NY (59,151)** follow closely, showcasing significant delay issues likely due to high traffic and operational challenges.
3. Airports like **Los Angeles, CA (30,230)** and **Houston, TX (33,067)** have comparatively lower total delay minutes but still make it to the top 10.
4. The average delay for these airports lies around **46,000 minutes**, highlighting a concentration of delays in major hubs that handle high volumes of air traffic.

This analysis provides valuable insights for identifying airports with the highest delays, helping stakeholders focus on improving infrastructure, managing congestion, and optimizing operational efficiency in these critical hubs.

### Flight Delay Insights: Trends, Contributors, and Top Airports

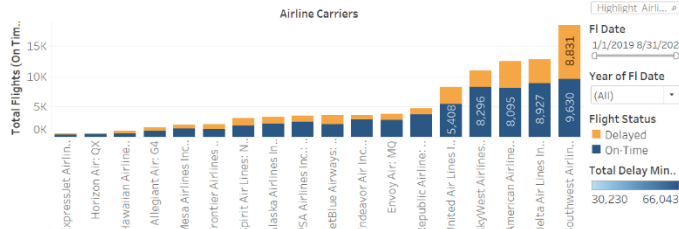
#### Top 10 Busiest Airports with Total Flight Delays

Highlighting the airports contributing the most to total delay minutes for better operational insights.



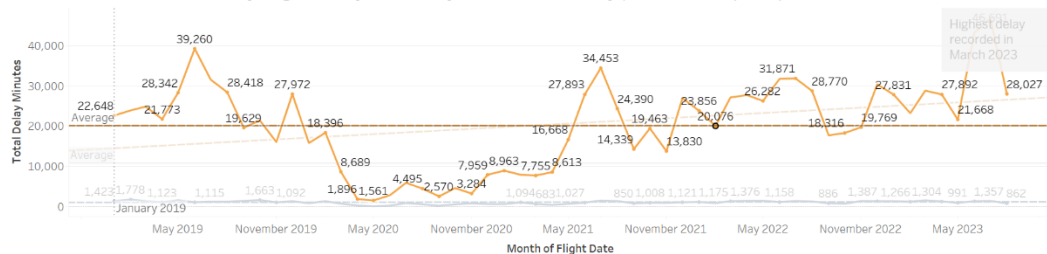
#### Delayed vs. On-Time Flights by Airline

Displaying the total flights operated and the share of delays for each airline.



#### Trends in Flight Delays Over Time

Analyzing monthly total delay minutes to identify patterns and peak periods.



Delayed vs On-Time Flights by A... Airline Delay Factors Breakdown Trends in Flight Delays Over Time Top 10 Busiest Airports with Tot... Flight Delay Insights

### Flight Delay Insights Dashboard

This dashboard highlights key trends and contributors to flight delays. The **Top 10 busiest airports**, such as **Chicago, Dallas/Fort Worth, and New York**, show the highest total delay minutes. Delays are also analyzed across airlines, with **Southwest Airlines** leading in total delays. Additionally, **monthly trends** reveal peak delays, notably in **March 2023**, providing a comprehensive view of delay patterns, enabling targeted improvements in airline operations and airport management.

## **Evaluation and Conclusion:**

The models performed decently in predicting flight delays, with the Gradient Boosting Regressor performing the best. The results indicate that machine learning can be a viable approach to predicting flight delays.

## **Future Work:**

Given more time, the models could be improved by tuning their hyperparameters. With better programming skills, more advanced techniques could be utilized such as ensemble methods or more complex neural networks. Better data, especially incorporating weather conditions, could also improve the models. If better functions/algorithms were available, they could be used to enhance model performance.

## **Future Enhancement Areas:**

The models could be improved by using techniques to handle the imbalance in the dataset. More features could be engineered from the existing data. The project could also be expanded to include real-time prediction of flight delays.

## **Project Deliverables:**

- Code: Python scripts for data preprocessing, modeling, and evaluation of flight delays.
- Database: Raw and processed datasets stored for further analysis and reproducibility.
- Visualization: Interactive Tableau dashboards to explore trends, delays, and insights.
- Report: Comprehensive documentation outlining methods, results, and key findings.
- PowerPoint Presentation: A concise summary highlighting project objectives, methodologies, results, and actionable insights