Nianyang Chen
198:439 Intro to Data Science
December 10, 2025

Project Final Report

1. Project Statement

Flight delays are a significant challenge in the aviation industry, costing the U.S. economy billions of dollars annually and affecting millions of passengers. This project attempts to build a flight delay prediction system that predicts whether commercial flights will experience significant delays (defined as >20% delay rate, i.e., at least 20% of flights are delayed).

Current flight tracking platforms primarily report delays after they occur rather than predicting them in advance. This reactive approach leaves travelers less able to make informed travel decisions and prevents airlines from proactively managing and planning their operations. This project addresses this gap by developing predictive models using historical flight data, airport congestion, and airline operational metrics. This project essentially aims to transform raw flight operations records from the past into understandable predictions that could benefit both passengers and airlines.

Concepts involved in this project include:

- Data Management by loading and managing large-scale flight datasets using Pandas.
- Data Visualization, where this project includes five visualizations for data analysis.
- Data Cleaning and Preprocessing by handling missing values, checking for potential negative values, and/or duplicate records.
- Feature Engineering, as this project created some categorical and interaction features from raw data.
- Machine Learning**,** where multiple algorithms (Logistic Regression, Random Forest, Gradient Boosting) are implemented and compared
- Model Evaluation in terms of model accuracy, confusion matrices, and error rates.

2. Links to - Demo video and GitHub (Public Access) :

Demo video link: Video Recording - Wed Dec 10 2025 21_15_05 GMT-0500 (Eastern Standard Time).mp4

Github Link: https://github.com/nychub/Flight-Delay-Prediction-Project-

3. Novelty and Importance:

As it has already been mentioned in the project statement, it is seemingly looks like that there are barely any consumer-facing tools that do predictive analytics accessible to everyday travelers for travel planning and making informed decisions, despite that flight delays could cost billions of dollars and affect millions of passengers each year. Hence, the importance of this project is to apply data science and machine learning techniques to this issue and create straightforward insights for travelvers for better travel decision-making.

Aspects that show the novelty of this project:

- Engineering Historical Aggregate Features that combine carrier, airport, and seasonal patterns.
- Conduct feature importance analysis to provide clear insights for both travelers and airlines
- Exposed the possibility of causing data leakage, which means that “a model uses information during training that wouldn't be available at the time of prediction” (IBM). In the case of this project, it means that I would need to ensure that my model uses only pre-flight information when I choose what features to use to implement a model.

4. Progress and Contribution - Also mention/cover individual contribution (if working a team) :

1. Import and load data:
   Dataset**:** Kaggle Airline Delay Dataset (Link)
   Dataset Information:
   - Have 171,666 rows of records and features 21 columns, including flight volumes, delay classifications by cause (carrier, weather, NAS, security, late aircraft), cancellations, and diversions rates
   - Includes both major and minor U.S. airlines and airports.
2. Data Visualizations and Analysis:
   I created a total of five data visualizations:
   - Overall Delay Distribution showing most flights have a delay rate between 0 and around 30%
   - Delay Rate by Airline: Top 15 airlines ranked from the highest delay rates
   - Delay Causes Pie Chart: showing carriers, late aircraft, and NAS are the significant factors to cause flight delays
   - Monthly Delay Trends: Clear seasonal patterns with peaks in summer (school break) and December (weather/holiday season)
   - Top Airports by Delay Rate: 20 airports with the highest delay rates identified
3. Data Cleaning:
   I conducted a few dataset checks about missing values, negative values, and duplicate rows, then handled missing values accordingly.
4. Feature Engineering:
   - Initially created:
     i. delay_rate: Percentage of flights delayed
     ii. cancellation_rate: Percentage of flights cancelled
     iii. diversion_rate: Percentage of flights diverted

        iv. season: Winter/Spring/Summer/Fall categorization
        v. peak_season: indicator for high-travel months (June, July, August, November, December)
    - As part of the Logistic Regression Improvement:
        i. Historical Aggregate Features
            1. carrier_avg_delay: Historical average delay rate by carrier
            2. airport_avg_delay: Historical average delay rate by airport
            3. month_avg_delay: Historical average delay rate by month
            4. Carrier_airport_avg_delay: historical average delay rate by carrier-airport combination
        ii. Interaction Features:
            1. peak_x_flights: Peak season multiplied by flight volume
            2. carrier_x_month: Carrier-month combination

5. Target Variable Creation:
    - I created a binary target variable for prediction based on the question of whether a flight would be significantly delayed. Accordingly, the threshold of the target variable would be set to greater than 20% which means that a flight has at least a 20% delay rate and is considered a significant delay.
    - Target Variable Distribution reveals that 62.1% of flights aren't significantly delayed, whereas 37.9% of flights are significantly delayed under this threshold.

I completed all of these works since it is an individual project.

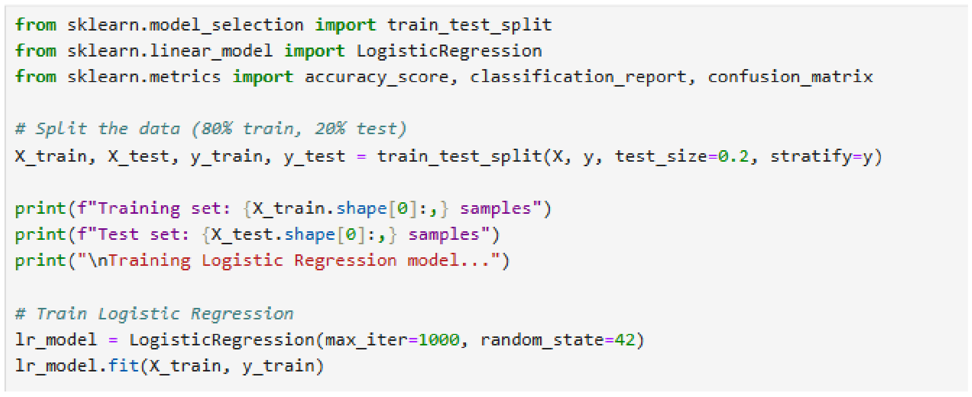
5. Models and Algorithms :

Split the data into a training set and a test set with 80% for training and 20% for testing.
Training set: 137,332 samples
Test set: 34,334 samples

Models implemented:

- Logistic Regression as a Baseline Model

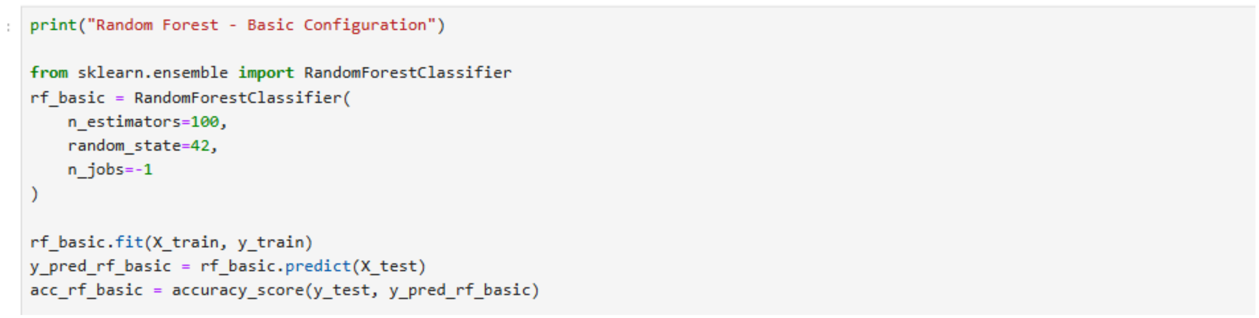
Part 6: Train Baseline Model (Logistic Regression)



```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Split the data (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)

print(f"Training set: {X_train.shape[0]:,} samples")
print(f"Test set: {X_test.shape[0]:,} samples")
print("\nTraining Logistic Regression model...")

# Train Logistic Regression
lr_model = LogisticRegression(max_iter=1000, random_state=42)
lr_model.fit(X_train, y_train)
```

- Random Forest

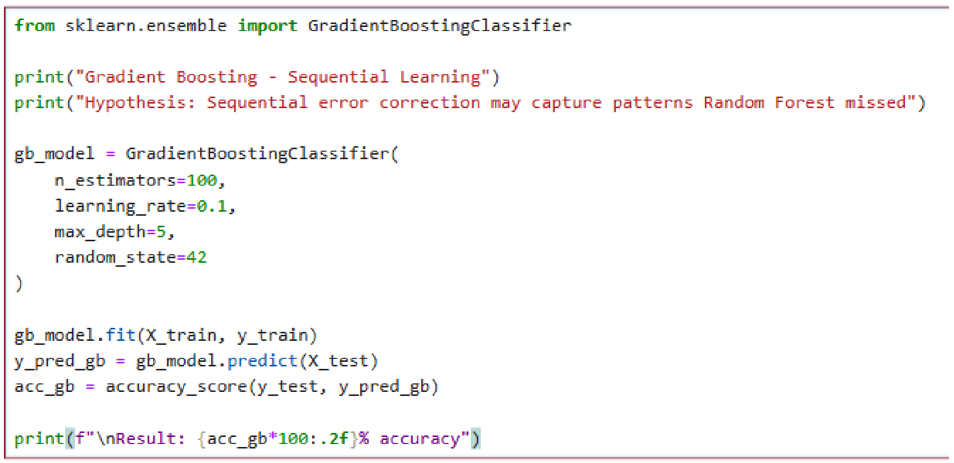
Part 7: Basic Random Forest



```python
print("Random Forest - Basic Configuration")

from sklearn.ensemble import RandomForestClassifier
rf_basic = RandomForestClassifier(
    n_estimators=100,
    random_state=42,
    n_jobs=-1
)

rf_basic.fit(X_train, y_train)
y_pred_rf_basic = rf_basic.predict(X_test)
acc_rf_basic = accuracy_score(y_test, y_pred_rf_basic)
```

- Gradient Boosting

Part 8: Try Gradient Boosting



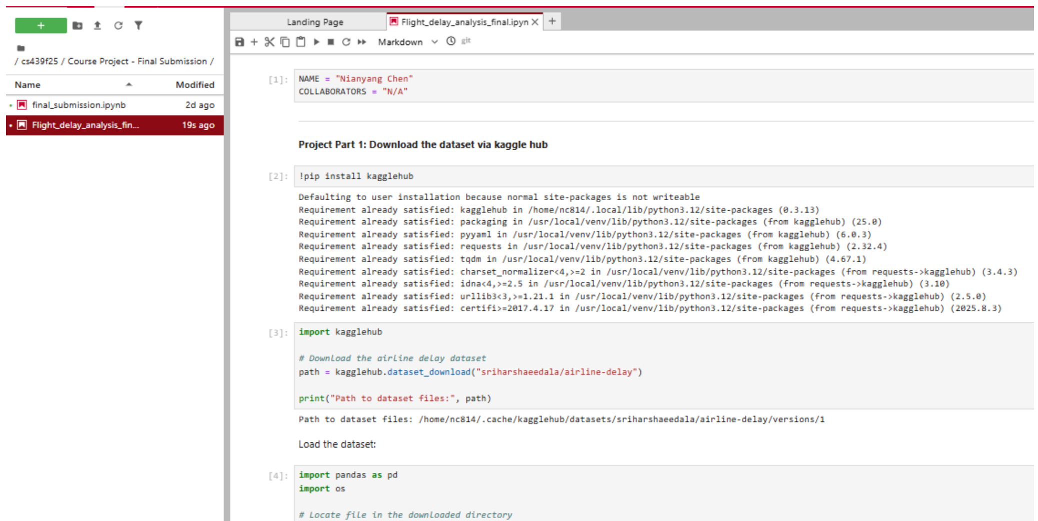
```python
from sklearn.ensemble import GradientBoostingClassifier

print("Gradient Boosting - Sequential Learning")
print("Hypothesis: Sequential error correction may capture patterns Random Forest missed")

gb_model = GradientBoostingClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=5,
    random_state=42
)

gb_model.fit(X_train, y_train)
y_pred_gb = gb_model.predict(X_test)
acc_gb = accuracy_score(y_test, y_pred_gb)

print(f"\nResult: {acc_gb*100:.2f}% accuracy")
```

6. Experimental Design :

Experiment	Model	Test Accuracy	Analysis	Results

1 (initial model implemented, featured in the progress report)	Logistic Regression	99.80%	The resulting accuracy is so high that it is "too good to be true." Then learned that it is an example of Data leakage	Fail and invalid
2	Logistic Regression	63.54%	It is an honest and realistic baseline model, but it fell short of the accuracy goal	Fail
3	Logistic Regression	70.57%	Adding Historical Features Improves Logistic Regression, but still a bit short of the goal.	Fail
4	Random Forest (Basic)	77.94%	Try a different model and is able to exceed the accuracy goal for the first time with a Basic Random Forest. Wonder if there is room for improvement.	Success
5	Random Forest with Tuning	78.49%	Tuning could boost Random Forest's Accuracy by testing different numbers of decision trees and tree depth.	Success - the best-performing model too
6	Random Forest with Balanced Class Weights	78.39%	This model confirms that the dataset is already reasonably balanced and that class weights are not critical.	Success
7	Gradient Boosting	77.50%	An alternative model to compare with Random Forest. See that Random Forest actually outperforms this model for this dataset	Success

7. Screenshots of Code and Outputs :


/ cs439f25 / Course Project - Final Submission /
Name
Modified
final_submission.ipynb
2d ago
Flight_delay_analysis_fin...
19s ago
Landing Page
Flight_delay_analysis_final.ipyn
Markdown

```
[1]: NAME = "Nianyang Chen"
     COLLABORATORS = "N/A"
```

Project Part 1: Download the dataset via kaggle hub

```
[2]: !pip install kagglehub

Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: kagglehub in /home/nc814/.local/lib/python3.12/site-packages (0.3.13)
Requirement already satisfied: packaging in /usr/local/venv/lib/python3.12/site-packages (from kagglehub) (25.0)
Requirement already satisfied: pyyaml in /usr/local/venv/lib/python3.12/site-packages (from kagglehub) (6.0.3)
Requirement already satisfied: requests in /usr/local/venv/lib/python3.12/site-packages (from kagglehub) (2.32.4)
Requirement already satisfied: tqdm in /usr/local/venv/lib/python3.12/site-packages (from kagglehub) (4.67.1)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/venv/lib/python3.12/site-packages (from requests->kagglehub) (3.4.3)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/venv/lib/python3.12/site-packages (from requests->kagglehub) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/venv/lib/python3.12/site-packages (from requests->kagglehub) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/venv/lib/python3.12/site-packages (from requests->kagglehub) (2025.8.3)
```

```
[3]: import kagglehub

     # Download the airline delay dataset
     path = kagglehub.dataset_download("sriharshaeedala/airline-delay")

     print("Path to dataset files:", path)

Path to dataset files: /home/nc814/.cache/kagglehub/datasets/sriharshaeedala/airline-delay/versions/1
```

Load the dataset:

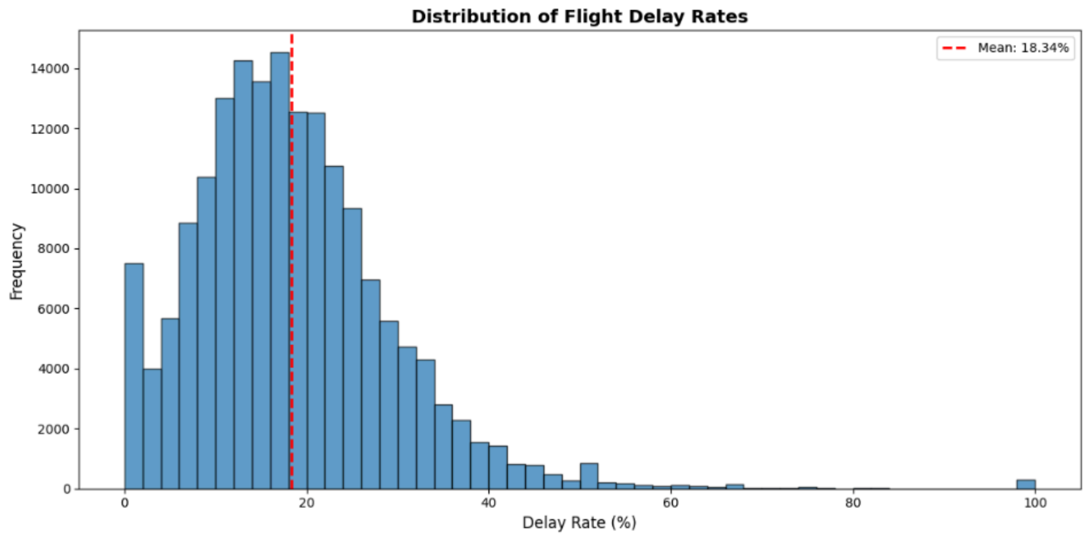
```
[4]: import pandas as pd
     import os

     # Locate file in the downloaded directory
```

Part 2: Dataset Visualizations:

```
[7]: import matplotlib.pyplot as plt
     import numpy as np
```

Visualization of Overall Delay Distribution:

```
[8]: # Create delay rate column
     df['delay_rate'] = (df['arr_del15'] / df['arr_flights']) * 100

     plt.figure(figsize=(12, 6))
     plt.hist(df['delay_rate'].dropna(), bins=50, edgecolor='black', alpha=0.7)
     plt.xlabel('Delay Rate (%)', fontsize=12)
     plt.ylabel('Frequency', fontsize=12)
     plt.title('Distribution of Flight Delay Rates', fontsize=14, fontweight='bold')
     plt.axvline(df['delay_rate'].mean(), color='red', linestyle='--', linewidth=2, label=f'Mean: {df["delay_rate"].mean():.2f}%')
     plt.legend()
     plt.tight_layout()
     plt.show()

     print(f"Average delay rate: {df['delay_rate'].mean():.2f}%")
     print(f"Median delay rate: {df['delay_rate'].median():.2f}%")
```



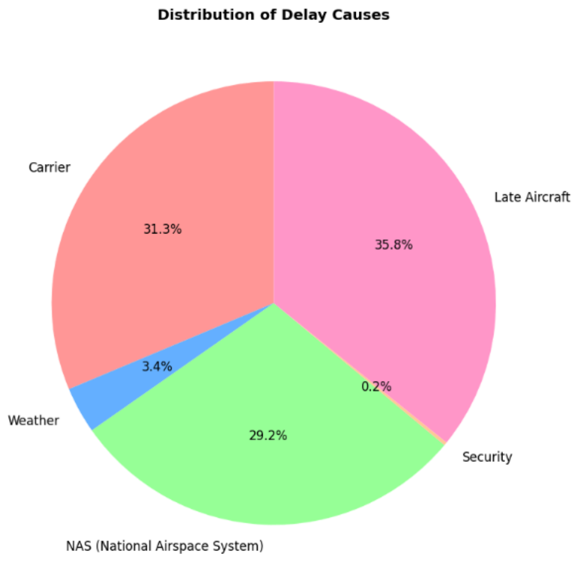
```
Average delay rate: 18.34%
Median delay rate: 17.07%
```

Visualization of Delay Rate by Airlines:

```
[ ]: # Calculate delay rate for each airlines
     carrier_delays = df.groupby('carrier_name').agg({'arr_flights': 'sum','arr_del15': 'sum'}).reset_index()

     carrier_delays['delay_rate'] = (carrier_delays['arr_del15'] / carrier_delays['arr_flights']) * 100
     carrier_delays = carrier_delays.sort_values('delay_rate', ascending=False).head(15)

     plt.figure(figsize=(14, 6))
     plt.barh(carrier_delays['carrier_name'], carrier_delays['delay_rate'], color='coral')
     plt.xlabel('Delay Rate (%)', fontsize=12)
     plt.ylabel('Airline', fontsize=12)
     plt.title('Top 15 Airlines by Delay Rate (% of flights delayed 15+ minutes)', fontsize=14, fontweight='bold')
     plt.gca().invert_yaxis()
     plt.tight_layout()
     plt.show()

     print(f"Average delay rate across all carriers: {(df['arr_del15'].sum() / df['arr_flights'].sum()) * 100:.2f}%")
```

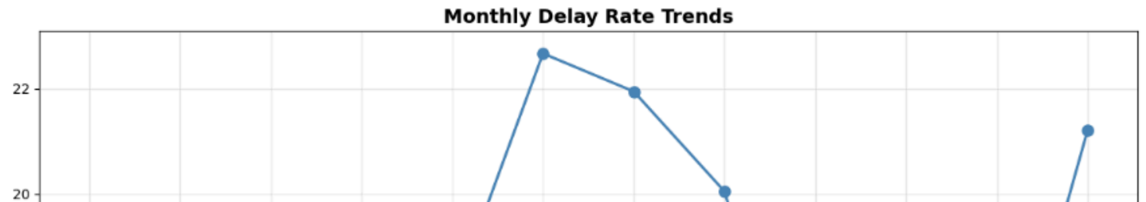
Visualization of Delay Causes:

```
[62]: # Sum up all delay causes
delay_causes = {'Carrier': df['carrier_ct'].sum(), 'Weather': df['weather_ct'].sum(), 'NAS (National Airspace System)': df['nas_ct'].sum(), 'Security': df['security_ct'].sum(), 'Late Aircraft': df['late_aircraft_ct'].sum()}

plt.figure(figsize=(10, 8))
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#ff99cc']
plt.pie(delay_causes.values(), labels=delay_causes.keys(), autopct='%1.1f%%', colors=colors, startangle=90, textprops={'fontsize': 12})
plt.title('Distribution of Delay Causes', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()

print("\nDelay Causes (Total Delayed Flights):")
for cause, count in delay_causes.items():
    print(f"{cause}: {count:,}")
```



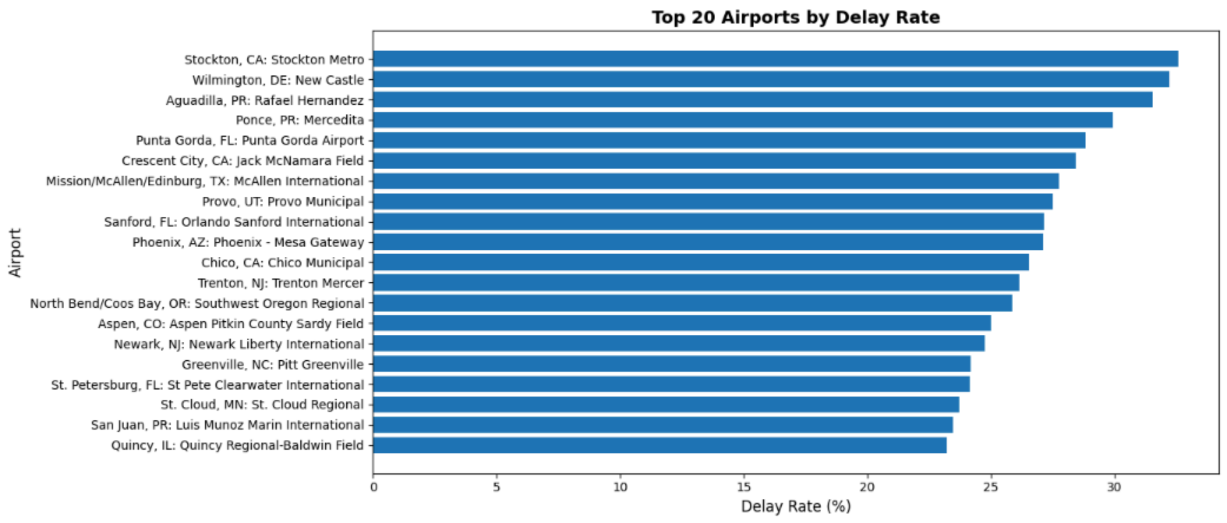
Visualization of Monthly Delay:

```
# Delays by month
monthly_delays = df.groupby('month').agg({'arr_flights': 'sum', 'arr_del15': 'sum'}).reset_index()

monthly_delays['delay_rate'] = (monthly_delays['arr_del15'] / monthly_delays['arr_flights']) * 100

plt.figure(figsize=(12, 6))
plt.plot(monthly_delays['month'], monthly_delays['delay_rate'], marker='o', linewidth=2, markersize=8, color='steelblue')
plt.xlabel('Month', fontsize=12)
plt.ylabel('Delay Rate (%)', fontsize=12)
plt.title('Monthly Delay Rate Trends', fontsize=14, fontweight='bold')
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```



Visualizate Top Airports by Delays

```
# Top airports by delay rate
airport_delays = df.groupby('airport_name').agg({
    'arr_flights': 'sum',
    'arr_del15': 'sum'
}).reset_index()

airport_delays['delay_rate'] = (airport_delays['arr_del15'] / airport_delays['arr_flights']) * 100
airport_delays = airport_delays[airport_delays['arr_flights'] > 1000]  # Filter for airports with significant traffic
airport_delays = airport_delays.sort_values('delay_rate', ascending=False).head(20)

plt.figure(figsize=(14, 6))
plt.barh(airport_delays['airport_name'], airport_delays['delay_rate'])
plt.xlabel('Delay Rate (%)', fontsize=12)
plt.ylabel('Airport', fontsize=12)
plt.title('Top 20 Airports by Delay Rate', fontsize=14, fontweight='bold')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



8. Detailed Analysis of Results and Key Results, and Evaluation :

Model Evaluation with Classification Metrics and Confusion Matrix:

Part 11: Evaluations on the best model (i.e. Random Forest with tuning)

```
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns

# Get predictions from best model (Random Forest with tuning)
y_pred_best = best_rf.predict(X_test)
y_pred_proba_best = best_rf.predict_proba(X_test)

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred_best, target_names=['On-Time', 'Delayed']))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred_best)
print(f"True Negatives (Correctly predicted On-Time):  {cm[0,0]:,}")
```

9. Conclusion

This project successfully developed a flight delay prediction system that exceeded the initial goal of 75% accuracy in the project proposal, achieving a 78.49% test accuracy using a Random Forest Model with Tuning. It can be seen through the feature importance analysis that past carrier-airport delay records are the dominant predictor of delays. Besides achieving technical goals, this project provides several benefits for both travelers and airlines:

Travelers could assess delay risk by checking historical delay rates for specific carrier-airport history before booking a flight. They could also make informed decisions for their travel plans, like trying to avoid peak travel months when possible, for a lower risk of delay. Lastly, they could consider alternative routes with better historical performance and have a lower delay rate over time.

Airlines should focus on operational improvements on historically problematic routes, address carrier-specific factors contributing to delays, and try to proactively allocate resources during peak travel periods, like crew scheduling and aircraft repositioning. They could also improve passenger communication with advanced delay warnings.

Ultimately, Flight delay prediction is a challenging task due to numerous factors (weather, mechanical issues, crew availability, etc.), but this project shows that machine learning can provide meaningful possibility predictions of a flight delay, which is beneficial for both travelers and airlines.

Reference:

https://www.kaggle.com/datasets/sriharshaeedala/airline-delay

http://ibm.com/think/topics/data-leakage-machine-learning