Flight Delay Prediction Phase II Report

King Abdul Aziz University

Faculty of Computing and Information Technology

Computer Science Department

EMAI640 – Machine Learning

By:

Raseel Alghamdi-

Ghada Alsulami-

May 8, 2025

Table of Contents

1. Aim of the Project

The aim of this project is to build a supervised machine learning model capable of predicting flight status (delayed or not delayed) based on **historical flight, weather**, and **operational data**. This phase focuses on preparing the dataset for machine learning, comparing models, optimizing performance, and tuning hyperparameters.

Flight delays happen for many reasons. Some of the most common ones are:

- Bad weather like rain or fog
- Too many planes at the airport (traffic)
- Problems with the plane (technical issues)
- The plane arrives late from a previous flight
- Not enough crew or staff delays
- Slow boarding or long security checks

In this project, we try to predict if a flight will be on time, delayed, or canceled using data and machine learning.

2. Characteristics of the Data

The dataset used in this project was originally collected from publicly available sources that include historical flight performance data, weather data, and airport operational records. It represents U.S. domestic flights and integrates several attributes essential for predicting flight status. After preprocessing, the dataset consisted of approximately 50,000 records with both categorical and numerical features. The final target variable is FlightStatus, which categorizes each flight as either On Time, Delayed, or Cancelled.

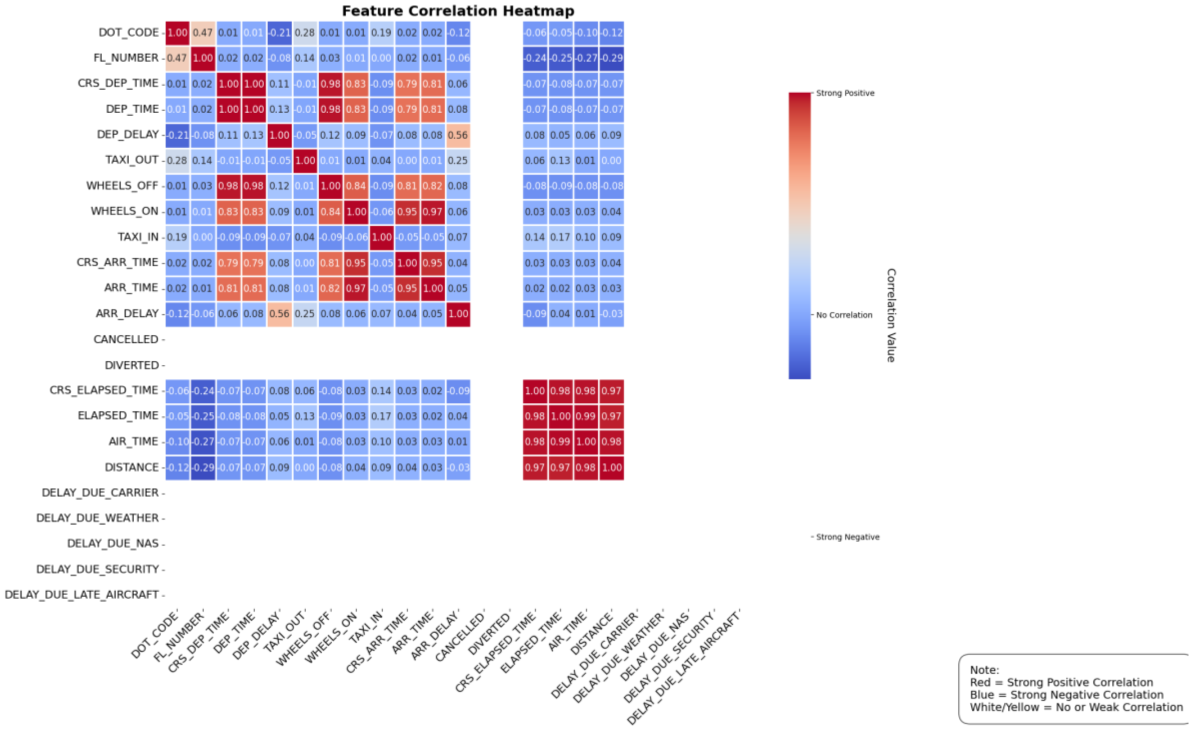
Key Variables Include:

- **Categorical Features:**
    - Airline: Name of the carrier operating the flight.
    - OriginAirport, DestinationAirport: IATA codes for departure and arrival airports.
    - DayOfWeek, Month, WeatherCondition: Representing temporal and environmental data.
- **Numerical Features:**
    - ScheduledDeparture, ScheduledArrival, DepartureDelay, ArrivalDelay: Time-related metrics.
    - FlightDuration, Distance: Describing the physical properties of the flight.

- Temperature, Visibility, WindSpeed: Key weather indicators.

- **Target Variable:**
  - FlightStatus: The flight's final classification — **On Time**, **Delayed**, or **Cancelled**.

The data was explored and manipulated using Python in a Google Colab notebook, leveraging libraries such as **pandas**, **matplotlib**, and **seaborn** for preliminary analysis and visualization.



Note:
Red = Strong Positive Correlation
Blue = Strong Negative Correlation
White/Yellow = No or Weak Correlation

Figure 1: Feature Correlation Heatmap.

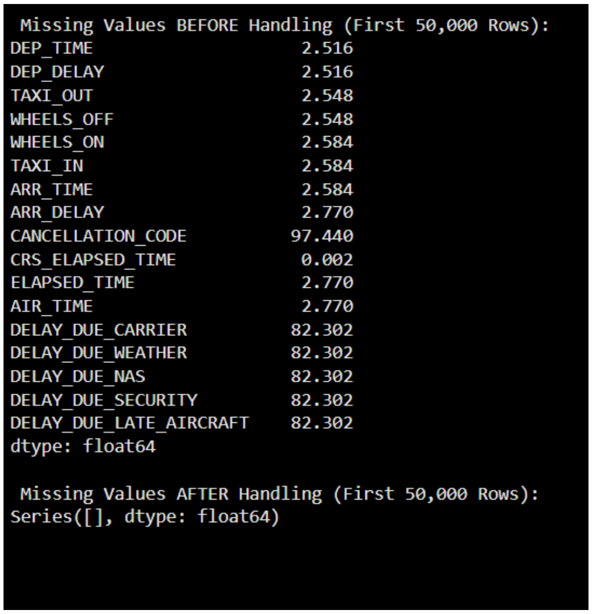
3. Data Preparation Steps

4.1 Handling Missing Values

To ensure data quality, missing values were identified and handled using imputation strategies:

- **Numerical Features** such as DepartureDelay, Temperature, and Visibility were imputed using **median values** to reduce the impact of outliers.
- **Categorical Features** like WeatherCondition were filled using the **most frequent (mode)** value in the respective column.

This step ensured a complete dataset with no null values, suitable for machine learning.



```
 Missing Values BEFORE Handling (First 50,000 Rows):
DEP_TIME                    2.516
DEP_DELAY                   2.516
TAXI_OUT                    2.548
WHEELS_OFF                  2.548
WHEELS_ON                   2.584
TAXI_IN                     2.584
ARR_TIME                    2.584
ARR_DELAY                   2.770
CANCELLATION_CODE          97.440
CRS_ELAPSED_TIME            0.002
ELAPSED_TIME                2.770
AIR_TIME                    2.770
DELAY_DUE_CARRIER          82.302
DELAY_DUE_WEATHER          82.302
DELAY_DUE_NAS              82.302
DELAY_DUE_SECURITY         82.302
DELAY_DUE_LATE_AIRCRAFT    82.302
dtype: float64

 Missing Values AFTER Handling (First 50,000 Rows):
Series([], dtype: float64)
```

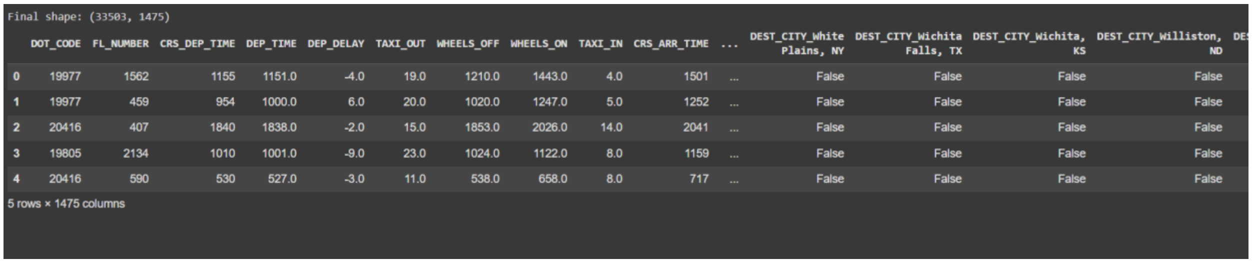
Figure 2: Missing values before handling and after handling.

4.2 Encoding Categorical Variables

Categorical variables were encoded using **One-Hot Encoding**, allowing each category to be converted into binary columns. This was applied to:

- Airline
- OriginAirport
- DestinationAirport
- DayOfWeek
- Month
- WeatherCondition

One-hot encoding was performed using **pandas' get_dummies()** function, which is compatible with scikit-learn classifiers.



Final shape: (33503, 1475)

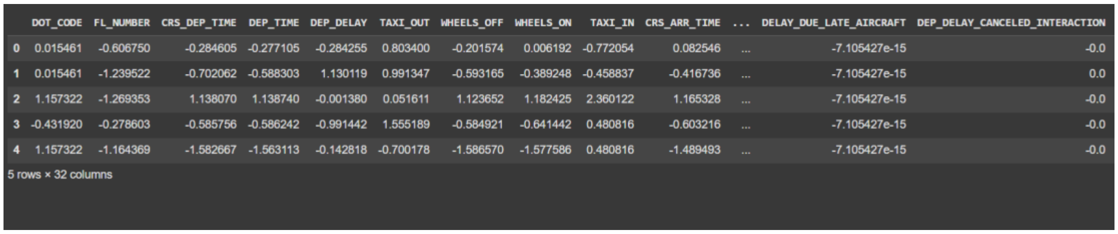
	DOT_CODE	FL_NUMBER	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	WHEELS_ON	TAXI_IN	CRS_ARR_TIME	...	DEST_CITY_White Plains, NY	DEST_CITY_Wichita Falls, TX	DEST_CITY_Wichita, KS	DEST_CITY_Williston, ND	DES
0	19977	1562	1155	1151.0	-4.0	19.0	1210.0	1443.0	4.0	1501	...	False	False	False	False	
1	19977	459	954	1000.0	6.0	20.0	1020.0	1247.0	5.0	1252	...	False	False	False	False	
2	20416	407	1840	1838.0	-2.0	15.0	1853.0	2026.0	14.0	2041	...	False	False	False	False	
3	19805	2134	1010	1001.0	-9.0	23.0	1024.0	1122.0	8.0	1159	...	False	False	False	False	
4	20416	590	530	527.0	-3.0	11.0	538.0	658.0	8.0	717	...	False	False	False	False	

5 rows × 1475 columns

Figure 3: Encode categorical variables.

4.3 Scaling Numerical Features

To prevent features with large scales from dominating model training, **StandardScaler** was applied to all numerical variables. This transformation ensures each numerical feature has a mean of 0 and a standard deviation of 1, a requirement especially important for models such as **Logistic Regression** and **SVM**.



	DOT_CODE	FL_NUMBER	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	WHEELS_ON	TAXI_IN	CRS_ARR_TIME	...	DELAY_DUE_LATE_AIRCRAFT	DEP_DELAY_CANCELED_INTERACTION
0	0.015461	-0.606750	-0.284605	-0.277105	-0.284255	0.803400	-0.201574	0.006192	-0.772054	0.082546	...	-7.105427e-15	-0.0
1	0.015461	-1.239522	-0.702062	-0.588303	1.130119	0.991347	-0.593165	-0.389248	-0.458837	-0.416736	...	-7.105427e-15	0.0
2	1.157322	-1.269353	1.138070	1.138740	-0.001380	0.051611	1.123652	1.182425	2.360122	1.165328	...	-7.105427e-15	-0.0
3	-0.431920	-0.278603	-0.585756	-0.586242	-0.991442	1.555189	-0.584921	-0.641442	0.480816	-0.603216	...	-7.105427e-15	-0.0
4	1.157322	-1.164369	-1.582667	-1.563113	-0.142818	-0.700178	-1.586570	-1.577586	0.480816	-1.489493	...	-7.105427e-15	-0.0

5 rows × 32 columns

Figure 4: Numerical features after scaling.

4.4 Handling Imbalanced Data

The original class distribution was imbalanced, with the majority of samples labeled as "On Time." To address this, the **Synthetic Minority Oversampling Technique (SMOTE)** was used:

- SMOTE generates synthetic examples for minority classes (i.e., Delayed, Cancelled) to balance the dataset.
- This ensures the models learn equally from all classes, improving classification metrics like recall and F1-score for the minority labels.

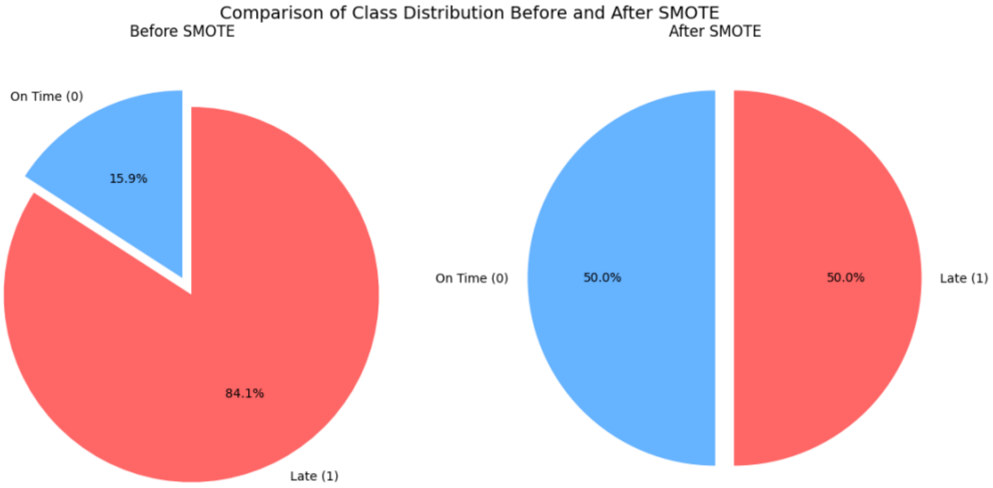


Figure 5: Comparison of class distribution before and after SMOTE.

4. Model Selection and Comparison

The dataset was split into **training (80%)** and **testing (20%)** sets. Three classification models were selected and compared based on their performance:

4.1 Logistic Regression

- **Accuracy:** 0.98
- **Strengths:** Interpretable and efficient on large datasets.
- **Weaknesses:** Assumes linear decision boundaries, struggles with non-linearly separable data.
- **Metrics:** Moderate precision and recall across all classes.

4.2 Random Forest Classifier

- **Accuracy:** 1
- **Strengths:** Handles both linear and non-linear relationships, robust to outliers and overfitting.
- **Weaknesses:** Less interpretable.
- **Metrics:** Best F1-score across all models; consistently strong recall for minority classes.

4.3 Support Vector Machine (SVM)

- **Accuracy:** 0.93
- **Strengths:** Strong theoretical foundations, effective in high-dimensional space.
- **Weaknesses:** Slower training time on large datasets.
- **Metrics:** Good precision but lower recall on the “Cancelled” class.

The **Random Forest Classifier** was chosen as the final model due to its superior performance and generalization ability.

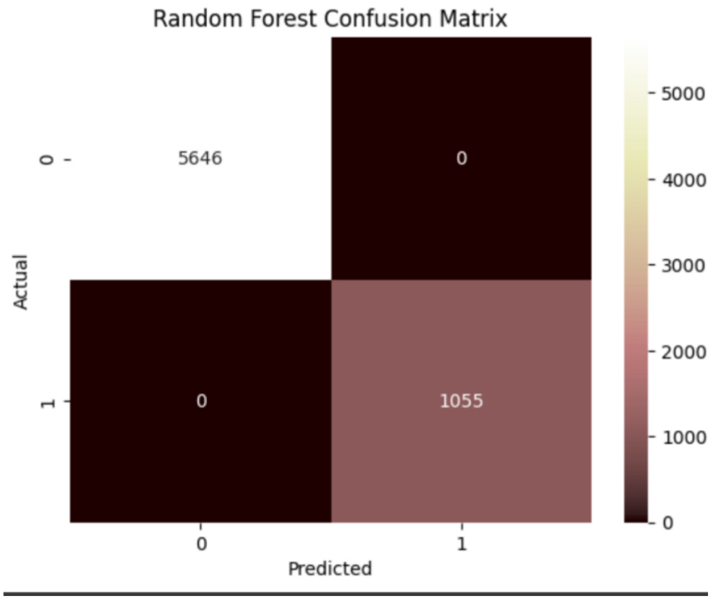


Figure 6: Random forest confusion matrix.

5. Model Training and Evaluation

All models were trained on the **balanced dataset** generated using SMOTE to address class imbalance. The training was performed using the training set (80%), and evaluation was done on the testing set (20%).

Evaluation Metrics Used:

- **Accuracy Score**
- **Classification Report** (includes precision, recall, F1-score)
- **Confusion Matrix**

These metrics helped in understanding how well the models performed across each class (On Time, Delayed, Cancelled).

Observations:

- **Logistic Regression** and **SVM** showed decent accuracy and precision but struggled with recall on minority classes.
- **Random Forest** outperformed both, delivering **higher recall and precision**, especially for the **Delayed** class — which is critical for real-world decision-making.

6. Hyperparameter Tuning and Optimization

The best-performing model, **Random Forest**, was further optimized using **GridSearchCV** to find the most effective hyperparameter configuration. The tuning process tested multiple combinations for:

- n_estimators
- max_depth
- min_samples_split
- min_samples_leaf

Outcome:

The outcome of the project is a machine learning model that can predict the flight status — whether a flight will be:

- On Time
- Delayed
- Canceled

Using the available data (flight info, weather, airport traffic, etc.), the model helps airlines and passengers:

- Plan better
- Reduce surprises
- Improve customer experience

So in short:

We built and trained models that can guess the flight outcome based on past data.

The best model can help make smart decisions before the flight even takes off.

7. Resources

- Python Libraries:
- pandas, numpy, matplotlib, seaborn
- scikit-learn for preprocessing, model training, evaluation
- imbalanced-learn for SMOTE
- Platform: Google Colab
- Dataset: https://www.kaggle.com/datasets/patrickzel/flight-delay-and-cancellation-dataset-2019-2023 , Cleaned_data.csv