Flights Delay Prediction using Machine Learning Models

BUAN 6341 / MIS 6341

Applied Machine Learning

Vijay Koju

Group 4
Abhishek Arya
Sai Divya Gudimella
Surya Madhuri Susarla
Venkateshwar Balakrishnan

Contents

Problem Statement :

In our project, we aim to develop a machine learning model for predicting flight delays in the airline industry. By leveraging historical flight data and relevant features such as weather conditions, airport congestion, and flight schedules, our goal is to create a predictive tool that can assist both airlines and passengers in anticipating and mitigating potential disruptions. This project addresses the critical need for proactive measures to enhance overall flight punctuality and passenger experience.

Data Description :

Sourced from Kaggle and originally associated with an IEEE transportation research paper, this dataset comprises 28,000 rows and 23 columns. The data spans flight departures from JFK airport between November 2019 and December 2020. The comprehensive dataset serves as a foundation for predicting flight delays, with key temporal attributes such as month, day of the week, and scheduled departure and arrival times. Essential carrier-related details, including the unique carrier code enrich the dataset. Weather conditions, encompassing temperature, humidity, wind speed, and pressure, play a pivotal role in comprehending external influences on flight delays. The dataset further incorporates noteworthy features such as departure delay, scheduled journey time, and distance, providing insights into operational aspects. With these diverse variables, including independent features and flight delay prediction as the target, the dataset is poised for the development of a robust machine learning model for accurate and proactive flight delay predictions.

Initial Data Preprocessing :

1. **Standardized Column Names**: Ensured consistency by renaming all columns for uniformity and clarity in the dataset.

2. **Deduplication**: Identified and removed any duplicate rows within the dataset to enhance data integrity.

3. **Column Selection** : Streamlined the dataset by excluding less relevant columns, such as day of the month, tail number, and taxi number, focusing on key features essential for our project.

4. **Data Type Optimization** : Adjusted data types of specific variables for better alignment with analytical requirements, optimizing memory usage and computational efficiency.

5. **Wind Direction Conversion**: Transformed cardinal directions of wind into degrees for a standardized representation, facilitating uniform analysis .

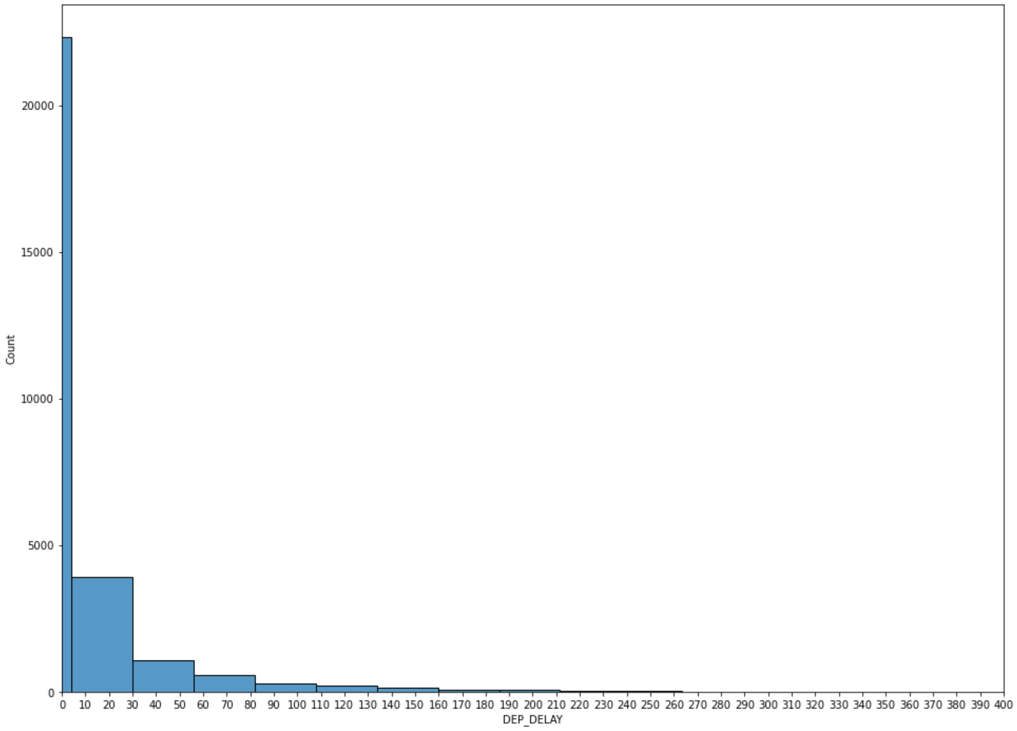
6. **Temporal Data Mapping**: Implemented data mapping for months, weekdays, and weekends, enhancing the interpretability and applicability of temporal information within the dataset.

7. **Target Variable Creation**: Introduced a new target variable, departure delay, by calculating the time difference between the actual departure time and the scheduled departure time .

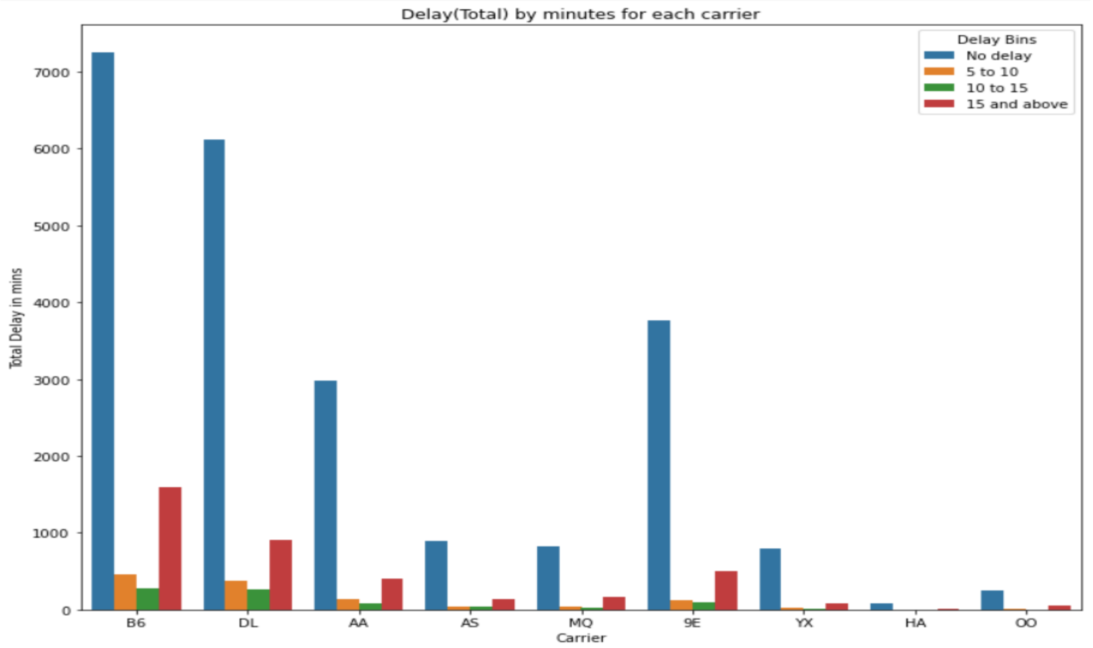
These preprocessing steps lay the foundation for a more refined and standardized dataset, setting the stage for subsequent exploratory data analysis and machine learning model development.

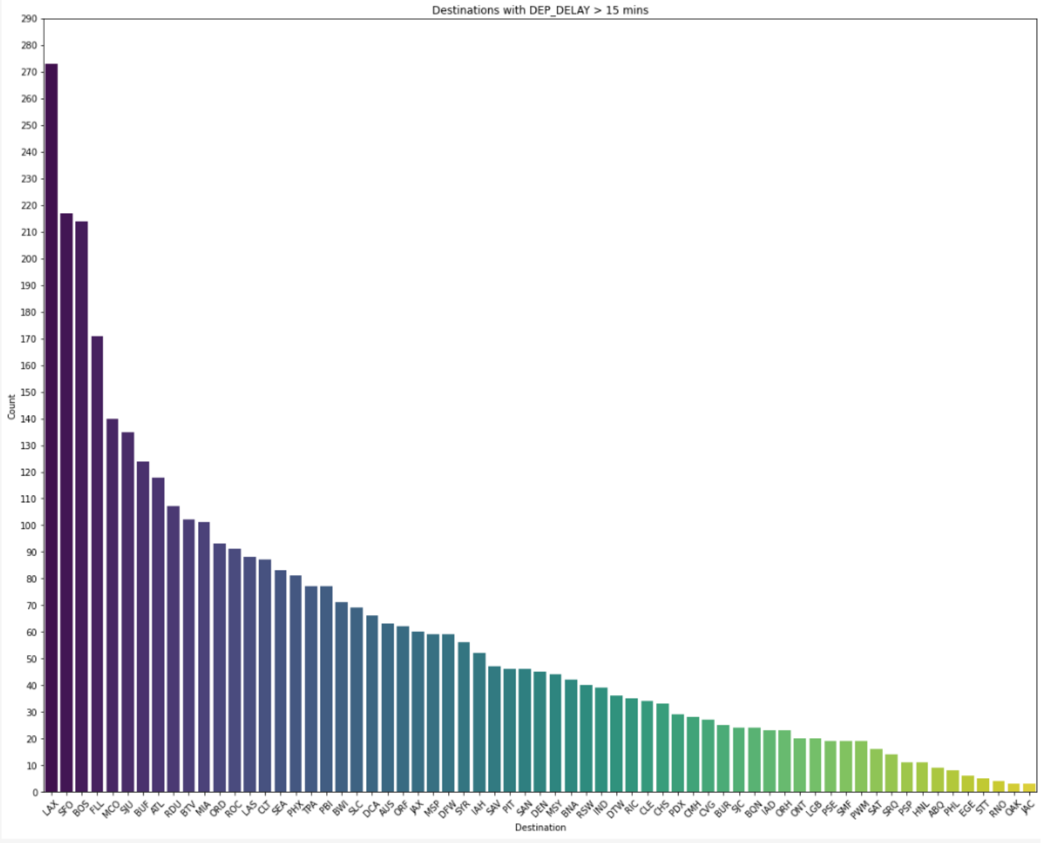
Exploratory Data Analysis :

The plot below illustrates the distribution of the target variable. Predominantly, the highest frequency of flight delays is observed within the 5-minute range, followed by a subsequent decrease in frequency within the 5 to 30-minute interval.

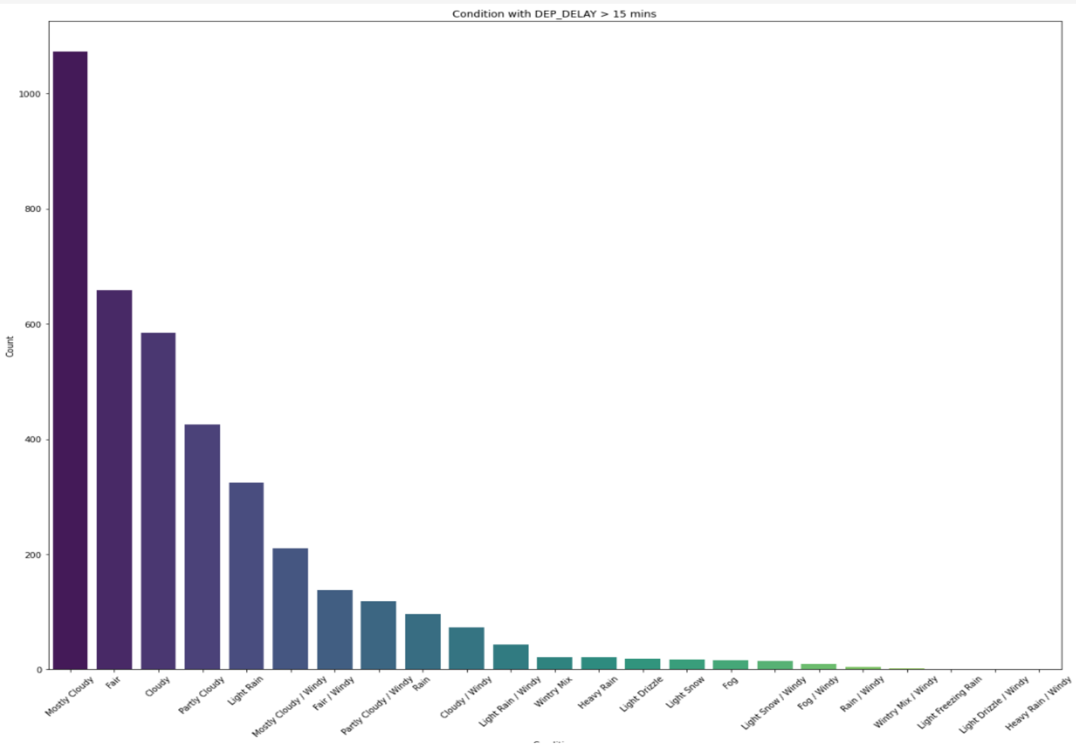


Below Graph depicts Departure Delay in Minutes for each carrier . It shows that Carrier B6 has highest number of Delays when compared to other flights



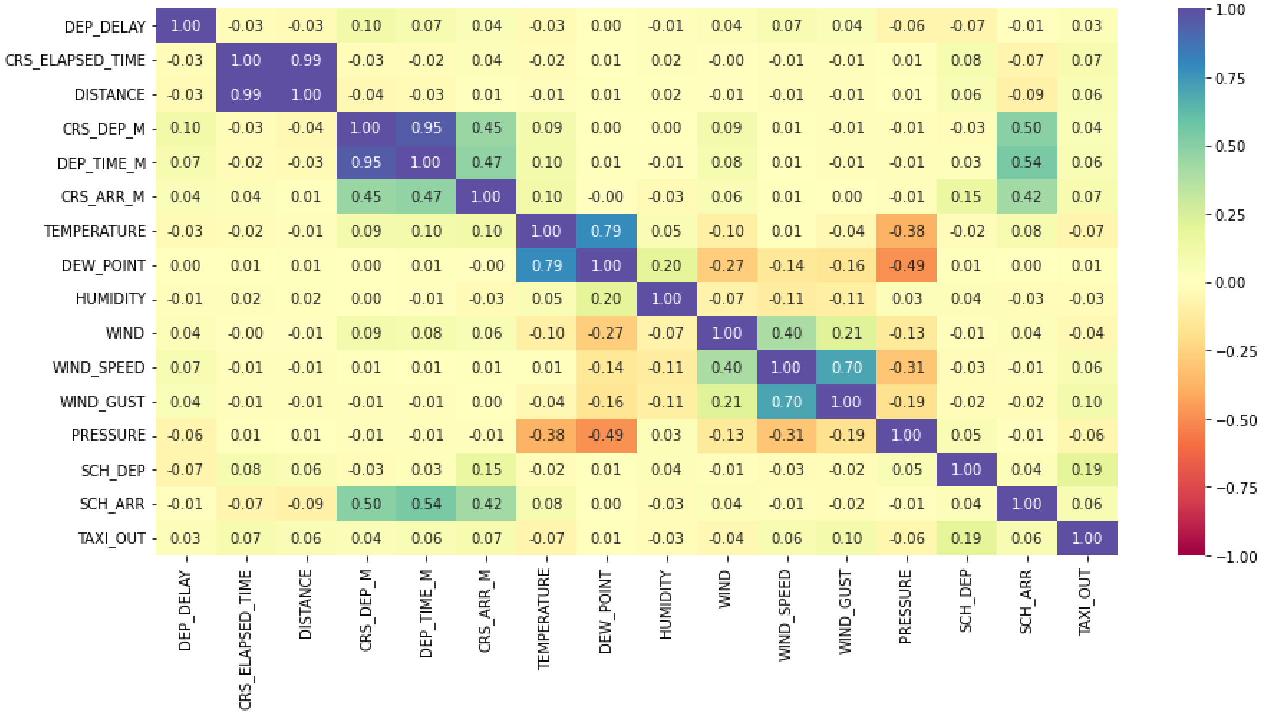


The visual representation above highlights that airports LAX, BOS, and SFO consistently exhibit the highest frequency of flight delays exceeding 15 minutes.



The above presented plot indicates that departure delays exceeding 15 minutes are predominantly associated with weather conditions categorized as "Mostly Cloudy."

Correlation Matrix :



	DEP_DELAY	CRS_ELAPSED_TIME	DISTANCE	CRS_DEP_M	DEP_TIME_M	CRS_ARR_M	TEMPERATURE	DEW_POINT	HUMIDITY	WIND	WIND_SPEED	WIND_GUST	PRESSURE	SCH_DEP	SCH_ARR	TAXI_OUT
DEP_DELAY	1.00	-0.03	-0.03	0.10	0.07	0.04	-0.03	0.00	-0.01	0.04	0.07	0.04	-0.06	-0.07	-0.01	0.03
CRS_ELAPSED_TIME	-0.03	1.00	0.99	-0.03	-0.02	0.04	-0.02	0.01	0.02	-0.00	-0.01	-0.01	0.01	0.08	-0.07	0.07
DISTANCE	-0.03	0.99	1.00	-0.04	-0.03	0.01	-0.01	0.01	0.02	-0.01	-0.01	-0.01	0.01	0.06	-0.09	0.06
CRS_DEP_M	0.10	-0.03	-0.04	1.00	0.95	0.45	0.09	0.00	0.00	0.09	0.01	-0.01	-0.01	-0.03	0.50	0.04
DEP_TIME_M	0.07	-0.02	-0.03	0.95	1.00	0.47	0.10	0.01	-0.01	0.08	0.01	-0.01	-0.01	0.03	0.54	0.06
CRS_ARR_M	0.04	0.04	0.01	0.45	0.47	1.00	0.10	-0.00	-0.03	0.06	0.01	0.00	-0.01	0.15	0.42	0.07
TEMPERATURE	-0.03	-0.02	-0.01	0.09	0.10	0.10	1.00	0.79	0.05	-0.10	0.01	-0.04	-0.38	-0.02	0.08	-0.07
DEW_POINT	0.00	0.01	0.01	0.00	0.01	-0.00	0.79	1.00	0.20	-0.27	-0.14	-0.16	-0.49	0.01	0.00	0.01
HUMIDITY	-0.01	0.02	0.02	0.00	-0.01	-0.03	0.05	0.20	1.00	-0.07	-0.11	-0.11	0.03	0.04	-0.03	-0.03
WIND	0.04	-0.00	-0.01	0.09	0.08	0.06	-0.10	-0.27	-0.07	1.00	0.40	0.21	-0.13	-0.01	0.04	-0.04
WIND_SPEED	0.07	-0.01	-0.01	0.01	0.01	0.01	0.01	-0.14	-0.11	0.40	1.00	0.70	-0.31	-0.03	-0.01	0.06
WIND_GUST	0.04	-0.01	-0.01	-0.01	-0.01	0.00	-0.04	-0.16	-0.11	0.21	0.70	1.00	-0.19	-0.02	-0.02	0.10
PRESSURE	-0.06	0.01	0.01	-0.01	-0.01	-0.01	-0.38	-0.49	0.03	-0.13	-0.31	-0.19	1.00	0.05	-0.01	-0.06
SCH_DEP	-0.07	0.08	0.06	-0.03	0.03	0.15	-0.02	0.01	0.04	-0.01	-0.03	-0.02	0.05	1.00	0.04	0.19
SCH_ARR	-0.01	-0.07	-0.09	0.50	0.54	0.42	0.08	0.00	-0.03	0.04	-0.01	-0.02	-0.01	0.04	1.00	0.06
TAXI_OUT	0.03	0.07	0.06	0.04	0.06	0.07	-0.07	0.01	-0.03	-0.04	0.06	0.10	-0.06	0.19	0.06	1.00

Based on the above correlated matrix we dropped highly correlated variables.

Data Processing using Column Transformers:

In our project, we've implemented a concise and efficient data preprocessing pipeline using scikit-learn' s Column Transformer. This pipeline addresses various preprocessing tasks, including missing value imputation, scaling, and encoding.

The `Col transformer` is configured with three main transformations:

1. **Imputation and Scaling**:

   For specific numerical features (`'CRS_ELAPSED_TIME'` and `'WIND_GUST'`), the pipeline applies a combination of logarithmic transformation, missing value imputation (commented out), and MinMax scaling.

2. **Scaling Only**:

   For a subset of numerical columns , the pipeline performs MinMax scaling without additional transformations.

3. **Imputation and One-Hot Encoding**:

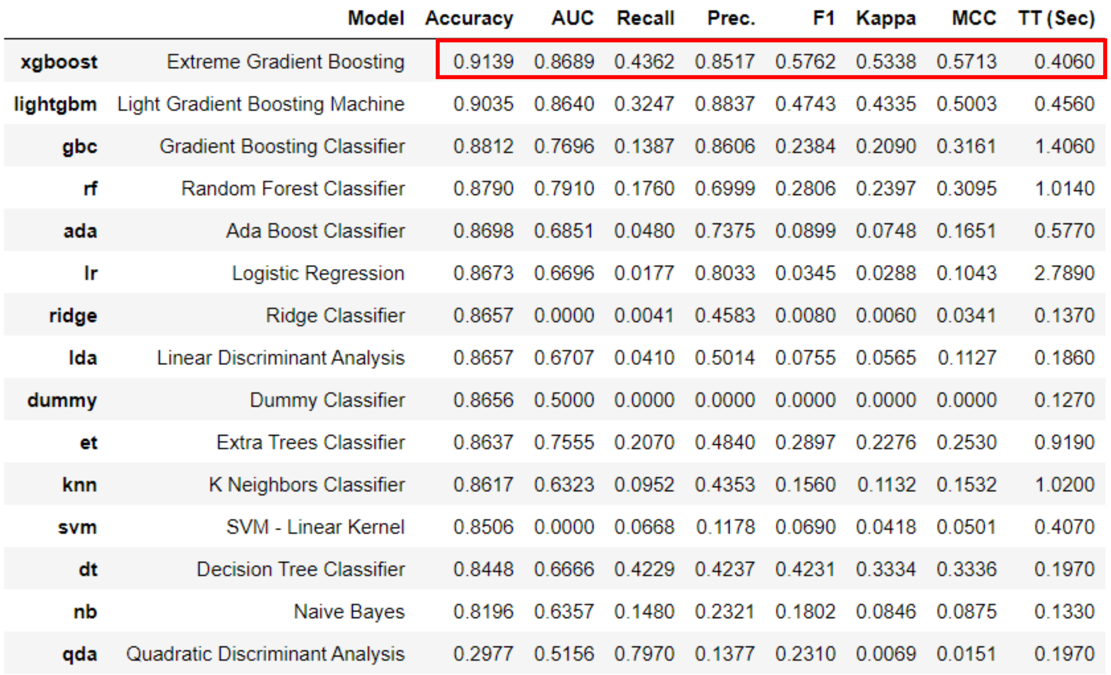
   For categorical columns , the pipeline combines simple imputation and one-hot encoding.

Machine Learning Models :

In our flight delay prediction project, we employed classification techniques to predict instances of delayed flights. Utilizing a variety of classification models , we leveraged historical flight data to train models that could discern patterns associated with delays. The objective was to categorize flights into delayed and non-delayed classes, providing valuable insights for proactive decision-making and resource allocation.

Using PyCaret:

In our efforts to predict flight delays, we are incorporating classification techniques. To kickstart the process, we utilized the PyCaret library to gain a comprehensive overview of how various machine learning models approach flight delay prediction.



	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.9139	0.8689	0.4362	0.8517	0.5762	0.5338	0.5713	0.4060
lightgbm	Light Gradient Boosting Machine	0.9035	0.8640	0.3247	0.8837	0.4743	0.4335	0.5003	0.4560
gbc	Gradient Boosting Classifier	0.8812	0.7696	0.1387	0.8606	0.2384	0.2090	0.3161	1.4060
rf	Random Forest Classifier	0.8790	0.7910	0.1760	0.6999	0.2806	0.2397	0.3095	1.0140
ada	Ada Boost Classifier	0.8698	0.6851	0.0480	0.7375	0.0899	0.0748	0.1651	0.5770
lr	Logistic Regression	0.8673	0.6696	0.0177	0.8033	0.0345	0.0288	0.1043	2.7890
ridge	Ridge Classifier	0.8657	0.0000	0.0041	0.4583	0.0080	0.0060	0.0341	0.1370
lda	Linear Discriminant Analysis	0.8657	0.6707	0.0410	0.5014	0.0755	0.0565	0.1127	0.1860
dummy	Dummy Classifier	0.8656	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1270
et	Extra Trees Classifier	0.8637	0.7555	0.2070	0.4840	0.2897	0.2276	0.2530	0.9190
knn	K Neighbors Classifier	0.8617	0.6323	0.0952	0.4353	0.1560	0.1132	0.1532	1.0200
svm	SVM - Linear Kernel	0.8506	0.0000	0.0668	0.1178	0.0690	0.0418	0.0501	0.4070
dt	Decision Tree Classifier	0.8448	0.6666	0.4229	0.4237	0.4231	0.3334	0.3336	0.1970
nb	Naive Bayes	0.8196	0.6357	0.1480	0.2321	0.1802	0.0846	0.0875	0.1330
qda	Quadratic Discriminant Analysis	0.2977	0.5156	0.7970	0.1377	0.2310	0.0069	0.0151	0.1970

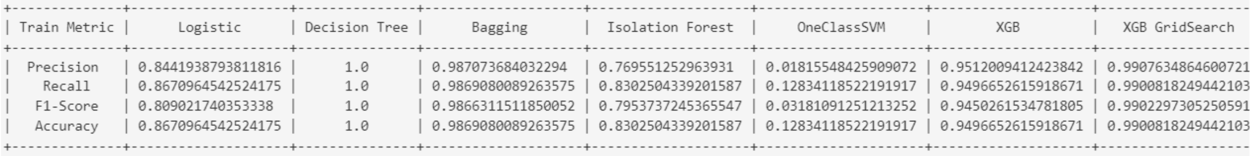
Based on the provided results, it is evident that XGBoost outperforms other models, exhibiting superior performance across multiple metrics such as accuracy, precision, AUC, and various others.

Classification Models :

To predict flight delays, we applied diverse classification techniques, such as Logistic Regression, Decision Trees, Bagging, Isolation Forest, One-Class SVM, and XGBoost. Following the model runs, as observed earlier, XGBoost emerged as the optimal choice based on evaluation metrics. To enhance performance, we incorporated hyperparameter tuning for XGBoost through GridCV.

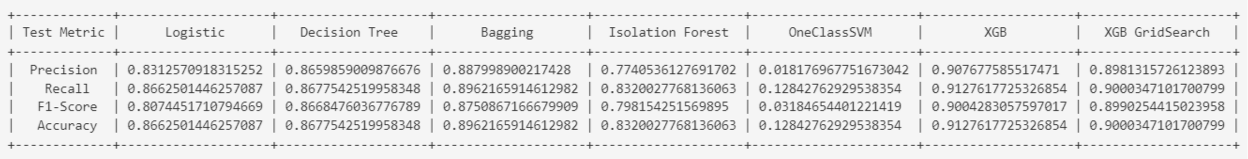
Results :

Evaluation Metrics for Training Data:



```
+--------------+--------------------+---------------+--------------------+--------------------+----------------------+--------------------+-------------------
| Train Metric |      Logistic      | Decision Tree |      Bagging       |  Isolation Forest  |     OneClassSVM      |        XGB         |  XGB GridSearch
+--------------+--------------------+---------------+--------------------+--------------------+----------------------+--------------------+-------------------
|  Precision   | 0.8441938793811816 |      1.0      | 0.987073684032294  | 0.769551252963931  | 0.01815548425909072  | 0.9512009412423842 | 0.9907634864600721
|    Recall    | 0.8670964542524175 |      1.0      | 0.9869080089263575 | 0.8302504339201587 | 0.12834118522191917  | 0.9496652615918671 | 0.9900818249442103
|   F1-Score   | 0.809021740353338  |      1.0      | 0.9866311511850052 | 0.7953737245365547 | 0.03181091251213252  | 0.9450261534781805 | 0.9902297305250591
|   Accuracy   | 0.8670964542524175 |      1.0      | 0.9869080089263575 | 0.8302504339201587 | 0.12834118522191917  | 0.9496652615918671 | 0.9900818249442103
+--------------+--------------------+---------------+--------------------+--------------------+----------------------+--------------------+-------------------
```

Evaluation Metrics for Testing Data :

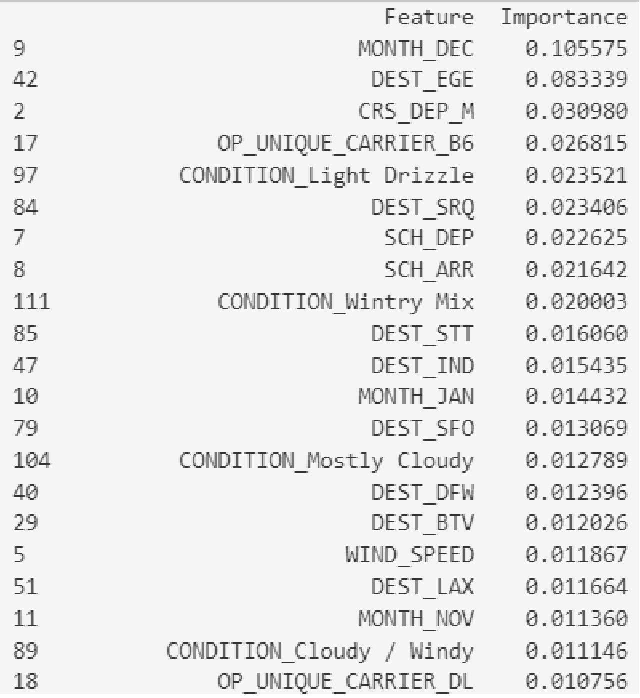


```
+-------------+--------------------+--------------------+--------------------+--------------------+----------------------+--------------------+--------------------+
| Test Metric |      Logistic      |   Decision Tree    |      Bagging       |  Isolation Forest  |     OneClassSVM      |        XGB         |   XGB GridSearch   |
+-------------+--------------------+--------------------+--------------------+--------------------+----------------------+--------------------+--------------------+
|  Precision  | 0.8312570918315252 | 0.8659859009876676 | 0.887998900217428  | 0.7740536127691702 | 0.018176967751673042 | 0.907677585517471  | 0.8981315726123893 |
|    Recall   | 0.8662501446257087 | 0.8677542519958348 | 0.8962165914612982 | 0.8320027768136063 | 0.12842762929538354  | 0.9127617725326854 | 0.9000347101700799 |
|   F1-Score  | 0.8074451710794669 | 0.8668476036776789 | 0.8750867166679909 | 0.798154251569895  | 0.03184654401221419  | 0.9004283057597017 | 0.8990254415023958 |
|   Accuracy  | 0.8662501446257087 | 0.8677542519958348 | 0.8962165914612982 | 0.8320027768136063 | 0.12842762929538354  | 0.9127617725326854 | 0.9000347101700799 |
+-------------+--------------------+--------------------+--------------------+--------------------+----------------------+--------------------+--------------------+
```

From the tables above, it is evident that XGBoost achieved a training accuracy of 99%, while its testing accuracy was recorded at 90%.

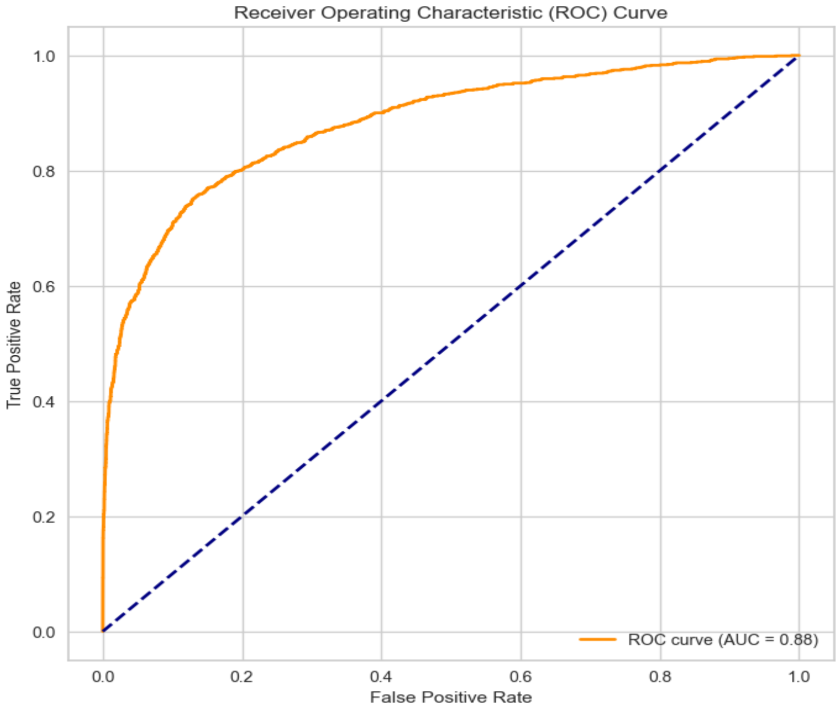
Feature Importance :

The features mentioned below impact our classification:

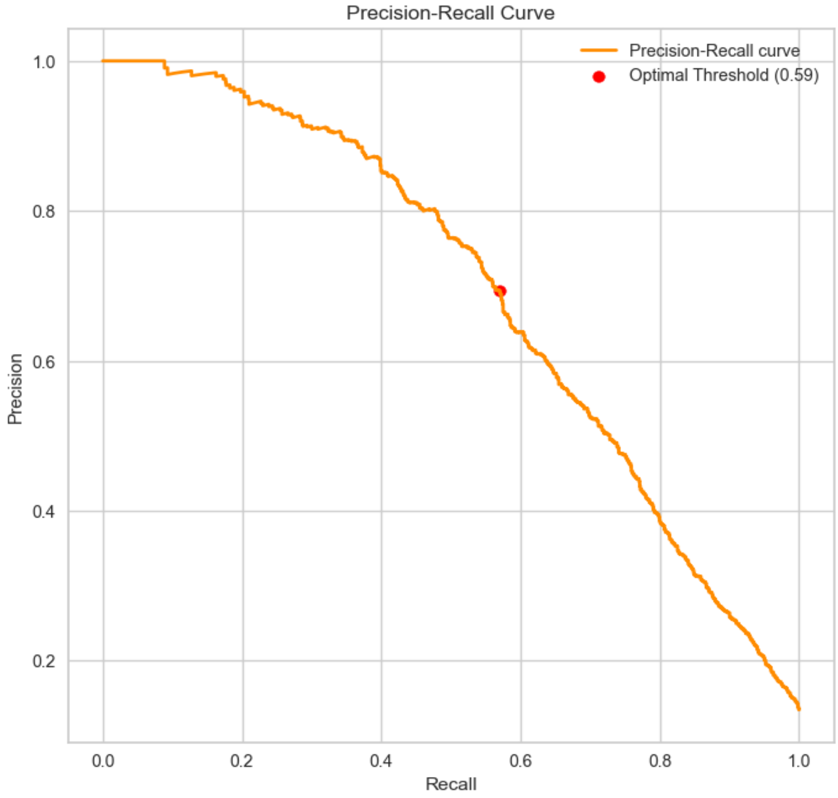


```
                          Feature  Importance
9                       MONTH_DEC    0.105575
42                       DEST_EGE    0.083339
2                       CRS_DEP_M    0.030980
17           OP_UNIQUE_CARRIER_B6    0.026815
97        CONDITION_Light Drizzle    0.023521
84                       DEST_SRQ    0.023406
7                         SCH_DEP    0.022625
8                         SCH_ARR    0.021642
111          CONDITION_Wintry Mix    0.020003
85                       DEST_STT    0.016060
47                       DEST_IND    0.015435
10                      MONTH_JAN    0.014432
79                       DEST_SFO    0.013069
104       CONDITION_Mostly Cloudy    0.012789
40                       DEST_DFW    0.012396
29                       DEST_BTV    0.012026
5                      WIND_SPEED    0.011867
51                       DEST_LAX    0.011664
11                      MONTH_NOV    0.011360
89       CONDITION_Cloudy / Windy    0.011146
18           OP_UNIQUE_CARRIER_DL    0.010756
```

ROC Curve for XG Boost Model:

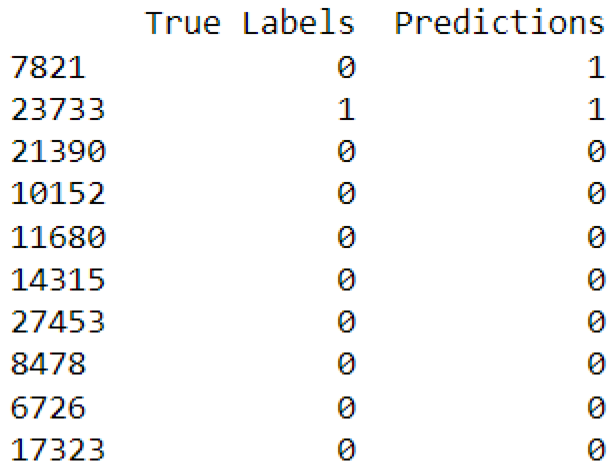


Precision Recall Curve for XG Boost Model:



Testing Unseen Data :

We further evaluated our model on unseen data, and it demonstrated strong performance by accurately predicting 9 out of 10 values .



```
       True Labels  Predictions
7821             0            1
23733            1            1
21390            0            0
10152            0            0
11680            0            0
14315            0            0
27453            0            0
8478             0            0
6726             0            0
17323            0            0
```